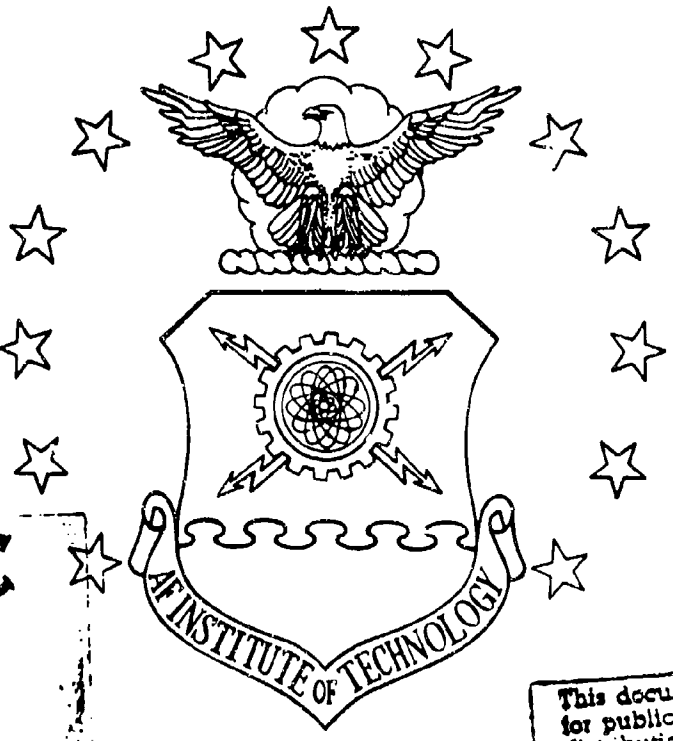


SOFTWARE COST ESTIMATING MODELS: A COMPARATIVE STUDY OF WHAT THE MODELS ESTIMATE

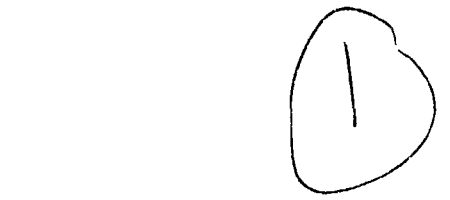
THESIS

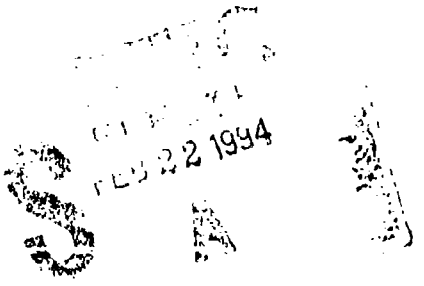
George A. Coggins, Captain, USAF
Roy C. Russell, Captain, USAF
AFIT/GCA/LAS/93S-4

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GCA/LAS/93S-4





SOFTWARE COST ESTIMATING MODELS: A COMPARATIVE STUDY OF WHAT THE MODELS ESTIMATE

THESIS

George A. Coggins, Captain, USAF
Roy C. Russell, Captain, USAF
AFIT/GCA/LAS/93S-4

Approved for public release; distribution unlimited

The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

AFIT/GCA/LAS/93S-4

SOFTWARE COST ESTIMATING MODELS:
A COMPARATIVE STUDY OF WHAT THE MODELS ESTIMATE

THESIS

Presented to the Faculty of the School of Logistics and Acquisition Management
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Cost Analysis

George A. Coggins, B.S.
Captain, USAF

Roy C. Russell, B.S.
Captain, USAF

September 1993

Acknowledgements

This research effort could not have been completed without the assistance of several key individuals. We would like to take this opportunity to acknowledge these people.

We would like to thank Ray Kile for his thoughtful insights regarding REVIC and Richard Maness of Martin Marietta for his assistance with SASET. Our thanks to Wellington Kao, Karen McRitchie, and Dan Gallorath of SEER Technologies for helping us understand SEER-SEM and to Jim Otte of PRICE Systems for his help with PRICE-S. It would have been impossible to complete this document without their assistance and patience in answering countless questions about their respective models. We thank our thesis advisor, Dan Ferens, for providing guidance and frequent feedback during this effort.

Finally, we would like to thank our wives and families for putting up with their "absentee husbands". Chris thanks his wife, Lara, and daughter, Chelsea, for their endless support and understanding. Andy thanks his wife, Debbie, and daughter, Jordan, for understanding the many times he couldn't be with them. We couldn't have done it without you.

Andy Coggins and Chris Russell

Table of Contents

List of Figures

List of Tables

Abstract

The objective of this thesis was to develop a consolidated document which highlights the differences in definitions, assumptions, and methodologies used by the REVIC, SASET, PRICE-S, and SEER-SEM cost models and examines the impact of these differences on the resulting estimates. To achieve this objective, the following research questions were investigated: (1) What differences exist between the cost models? (2) How do these differences impact the resulting cost estimates? (3) To what degree can we explain and adjust for differences between cost models?

Seven specific areas of model differences were addressed including: (1) software development phases, (2) development activities and cost elements, (3) source lines of code and language differences, (4) key model attributes and key cost drivers, (5) implications of project size on model output, (6) impact of schedule compression and extensions, and (7) distinctive characteristics of model data bases.

A hypothetical baseline test case was developed to determine if users could explain and adjust for known model differences. Although the researchers felt the differences could be partially explained, it was very difficult to logically adjust for the differences. It is the researchers' opinion that the underlying equations and model assumptions are so dissimilar that objective normalization efforts are virtually impossible for the average model user.

SOFTWARE COST ESTIMATING MODELS:
A COMPARATIVE STUDY OF WHAT THE MODELS ESTIMATE

I. Introduction

General Issue

The computer revolution has dramatically impacted all facets of society. From manned space flight to Nintendo™ games, computers have become an integral part of daily life. We use computers to write term papers, analyze properties of new drugs, and navigate aircraft. Most children gain basic computer skills in elementary school through interactive learning sessions and on-line games.

However, within the Department of Defense (DoD), computer applications go far beyond games. The basic role of the military is to defend and protect the national interests of the United States. In some instances, existing war fighting capability cannot adequately address perceived threats. When this occurs, the military community may initiate a new weapons system acquisition.

Faced with rapidly changing technology and shifting priorities, new weapon systems are increasingly dependent on computers and associated software. As shown in Figure 1-1, current trends indicate computer software development and maintenance costs for military systems generally exceed those of the hardware system (1:18).

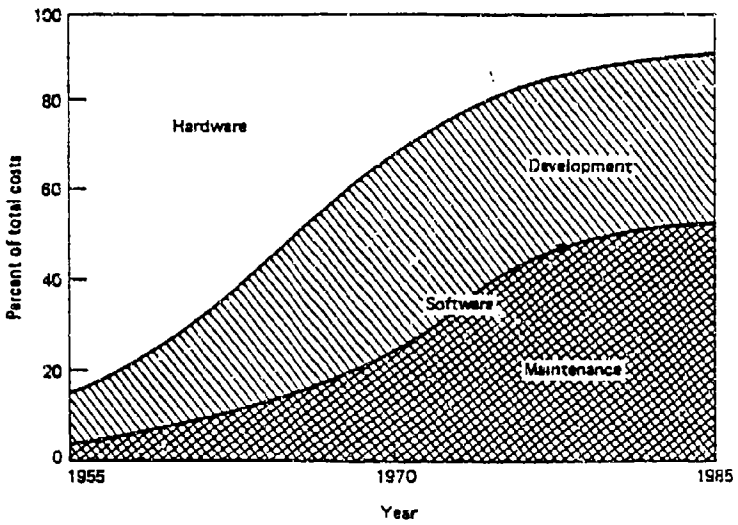


Figure 1-1. Distribution of Software and Hardware Costs (1:18)

As new military applications of computers were discovered, software acquisition costs within the DoD skyrocketed. Software costs increased from approximately $3.3 billion in 1974 to $10 billion in 1984 (2:1462). This rapid growth in software costs shows no signs of abating. A recent study of DoD mission critical software costs predicts a 12 percent annual growth rate from $11.4 billion in 1985 to $36 billion in 1995 (2:1462). Given the current environment of spiraling software costs and tight budget constraints, the need for accurate estimates cannot be understated.

Unfortunately, software cost estimating capabilities have not kept pace with the explosive growth of software development. Parametric software cost models used in the early 1980s were accurate to within 20 percent of the project's actual cost only 68% of the time (1:495); however, no evidence of significant improvements in cost estimation models was noted in several recent studies of parametric cost models (3, 4, 5, 6).

This does not imply cost analysts should avoid using parametric cost models. To the contrary, in the early stages of many projects when little actual cost data is available, parametric cost models are ideal for generating initial "ballpark" estimates. However, users should realize all new projects have a certain degree of inherent risk and no model can estimate the cost of all possible contingencies for all projects. Even if such a model was available, analysts are not automatically guaranteed good estimates. A software cost estimating model, like any other computer-based model, "is a 'garbage in - garbage out' device: if you put poor sizing and attribute-rating data in on one side, you will receive poor cost estimates out the other side" (1:308).

Cost analysts have a variety of parametric models available to aid preparation of software cost and schedule estimates. Currently, the four Air Force preferred software cost estimating models are: REVIC, SASET, PRICE-S and SEER-SEM (7). Each model has its own terminology, assumptions, estimating methodologies, strengths, and limitations. A thorough understanding of the models and their differences is essential for accurate, credible estimates.

Specific Problem

No single or consolidated document exists which explains the technical characteristics of the four preferred software cost models. As a result, it is difficult to evaluate and compare cost estimates generated by different cost models. Analysts must refer to numerous users manuals, contact software model vendors, or rely on second-hand information when preparing and comparing estimates from different models. In many instances, existing documentation does not fully explain the intricacies of the models. More

importantly, many users are unaware of the differences between models or what impact these differences have on the resulting estimates.

Research Objective

The objective of this thesis is to develop a consolidated document which highlights the differences in definitions, assumptions, and methodologies used by the REVIC, SASET, PRICE-S, and SEER-SEM cost models and examines the impact of these differences on the resulting estimates. To achieve this objective, the following questions must be investigated:

(1) What differences exist between the cost models?

(2) How do these differences impact the resulting cost estimates?

(3) To what degree can we explain and adjust for differences between cost models?

Scope of Research

This research effort was undertaken to support the Air Force Cost Analysis Agency (AFCAA). Specifically, AFCAA requested a technical analysis of the REVIC, SASET, PRICE-S, and SEER-SEM software cost estimating models. As a result, only these four models were chosen for analysis. No research effort was expended researching other existing cost estimating models such as SLIM, COSTMODL, System-4, or Checkpoint.

This effort did not specifically research the estimating accuracy between models. Research was conducted with the intent of explaining the differences between the models and examining the impact of these differences on the resulting estimates. Cost analysts should consider the strengths and weaknesses of each model as well as availability of information, time

constraints, and the nature of the proposed project prior to selecting a specific model.

Definitions

The following definitions are provided to ensure the reader and researchers have a mutual understanding of key terms and concepts used in this thesis.

Algorithm. A mathematical set of ordered steps leading to the optimal solution of a problem in a finite number of operations (8:557).

Analogy. An estimating methodology that compares the proposed system to similar, existing systems. After adjusting for complexity, technical, or physical differences, the cost estimator extrapolates the estimated cost for the new system using cost data from the existing system (9:A-5).

CSCI, CSC, and CSU. Large software development efforts are generally broken down into smaller, more manageable entities called computer software configuration items (CSCIs). Each CSCI may be further broken down into computer system components (CSCs) and each CSC may be further broken down into computer software units (CSUs) (10:B-14).

Cost Estimating. "The art of collecting and scientifically studying costs and related information on current and past activities as a basis for projecting costs as an input to the decision process for a future activity." (11).

Cost Model. A tool consisting of one or more cost estimating relationships, estimating methodologies, or estimating techniques and used to predict the cost of a system or its components (9:A-23).

Expert Opinion. An estimating methodology which queries technical experts and users regarding the estimated cost of a proposed system (8:581).

Hardware. Consists of the physical and electrical components of a computer system including items such as circuits, disk drives, wiring, and associated peripherals involved in the actual functions of the computer (12).

Normalization. The process of rendering constant or adjusting for known differences (8:594).

Parametric Cost Model. A model that employs one or more cost estimating relationships for measurement of costs associated with the development of an item based on the project's technical, physical, or other characteristics (8:596).

PRICE-S. Programmed Review of Information for Costing and Evaluation - Software. A commercial software cost estimating model distributed by PRICE Systems, a division of Martin Marietta Corporation. See the Cost Model Description section in Chapter II for details regarding this model.

REVIC. Revised Enhanced Version of Intermediate COCOMO. A non-proprietary parametric cost model used to estimate software costs. See the Cost Model Description section in Chapter II for details regarding this model.

SASET. Software Architecture Sizing and Estimation Tool. A non-proprietary parametric cost model used to estimate software costs. See the Cost Model Description section in Chapter II for details regarding this model.

SEER-SEM. System Evaluation and Estimation of Resources - Software Estimation Model. A commercial software cost estimating model developed by Galorath Associates, Incorporated. SEER-SEM is currently site-licensed for Air Force use. See the Cost Model Description section in Chapter II for details regarding this model.

Software. The combination of computer programs, data, and documentation which enable computer equipment to perform computational or central functions (12).

Software Development Cycle. The software development cycle is typically broken into 8 phases: (1) System Requirements Analysis and Design, (2) Software Requirements Analysis, (3) Preliminary Design, (4) Detailed Design, (5) Code and CSU Testing, (6) CSC Integration and Testing, (7) CSCI Testing, and (8) System Testing. Software maintenance is often considered the ninth phase in this sequence (10).

Software Maintenance. Software does not break or wear out; maintenance refers to corrective, adaptive, and perfective changes to software. Changes result when correcting software errors (corrective), responding to changing data or processing requirements (adaptive) and improving features through enhancements (perfective) (13:4).

Source Line of Code (SLOC). For purposes of this research effort, SLOC is defined as all lines of executable and non-executable code with the exception of embedded comments. See Chapter IV, Findings, for specific model definitions for SLOC.

Thesis Structure

The remainder of this research effort is directed at answering the investigative questions. The information gained by answering these questions will allow the researchers to compile a consolidated document which highlights the differences between the cost models and examines how these differences impact the cost estimates. Chapter II, Literature Review, reviews recent publications in the area of software cost estimating and describes each of the

cost models selected for review. Chapter III, Methodology, explains how the research effort was structured to gather information needed to answer the investigative questions. Chapter IV, Findings, analyzes the information obtained and answers the investigative questions. Chapter V, Conclusions and Recommendations, draws an overall conclusion regarding the differences between cost models based on the literature review and information obtained and analyzed in the preceding sections of the thesis. Chapter V also identifies areas where further research may be warranted.

II. Literature Review

Overview

This chapter reviews recent publications and research efforts in the field of software cost and schedule estimation. Specifically, this review (1) examines software development issues which impact the accuracy of cost estimates, (2) explains a normalization technique for comparing estimates generated by different cost models, and (3) provides descriptions of the cost models reviewed in this research effort.

Software Development Issues

Software developers and cost estimators seldom use the phrase "on-time and under-budget" when describing their latest software project. Three software development issues contribute to this problem: ever-increasing project size and complexity, software sizing problems, and unstable software requirements and specifications.

Ever-Increasing Size and Complexity. Since the beginning of software development, there has been a race between the capabilities of the tools available to programmers and the increasing complexities and sizes of the programs they were called upon to create (14:6). Early programming languages, such as Assembly language, required the programmer to transform a problem from its mathematical form into the step-by-step format demanded by the language. Assembly language programming was a slow, time-consuming method of generating code. However, high ordered languages (HOL) such as FORTRAN and Ada increased programmer efficiency because each line of HOL generated several Assembly commands.

As programming efficiency increased, the size of software programs also increased. Table 2-1 provides examples of software development efforts for several recent projects.

Table 2-1. Examples of Software Content in Recent Projects (15:100, 101, 104)

Project Description	Lines of Code	Labor (Manyears)	Development Cost ($ millions)
1989 Lincoln Continental	83,517	35	1.8
Lotus 1-2-3 v. 3	400,000	263	7.0
Citibank Autoteller	780,000	150	13.2
Space Shuttle	25,600,000	22,096	1200.0

As the preceding table indicates, software programming has become a more intricate, and costly, component of major projects.

One apparent solution for tackling large projects is to hire more programmers. However, empirical evidence indicates this approach is seldom applicable when dealing with software projects (16:18). Frederick Brooks, author of The Mythical Manmonth, notes software programming is not a perfectly partitionable task — simply adding people does not guarantee the job will be accomplished sooner (16:18). Although total effort initially decreases as workers are added, total effort actually increases as more and more workers are added to a project. Brooks attributes this effect to increased intercommunication needs (16:18). As more workers are added, more intercommunication is necessary to ensure all the workers work toward the same goal. Beyond a certain point, the benefits gained by adding more workers are outweighed by the increased communication needs (16:19). Figure 2 -1 illustrates Brooks' Law.

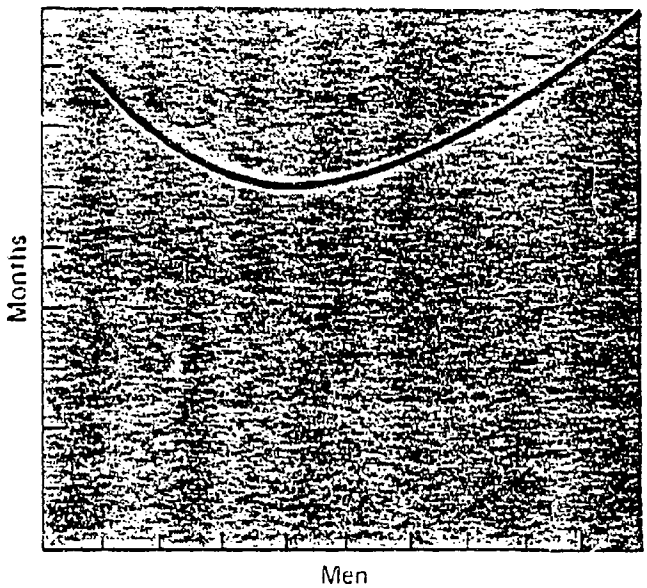


Figure 2-1. Effect of Adding Personnel to a Software Project (16:19)

Not only are projects becoming larger, but project complexity is also increasing. In his article "No Silver Bullet: Essence and Accidents of Software Engineering", Brooks states:

> Software entities are more complex for their size than perhaps any other human construct because no two parts are alike (at least above the statement level) ... In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound. (17:11)

Software Sizing Problems. Estimating project size has been described as a very difficult undertaking "often considered to require black magic" (18:19). To measure size, one requires a measurement technique. Although a number of sizing methodologies exist, two of the most commonly used methods are counting source lines of code (SLOC) and using function point algorithms (or some variation of the basic function point algorithm).

One of the most common techniques for measuring project size is counting the number of SLOC. Many software cost estimating models assume there is a relationship between SLOC developed and estimated development cost (1:58; 14:25; 19:5-2; 20:1). This relationship appeals to the layman since it seems logical for development costs to increase as project size increases. (Note: These models also take many other factors into account when generating cost estimates. See the Cost Model Description section in this chapter for additional information on each model).

Using lines of code as a sizing technique presents an unusual paradox. Most cost models require an estimated number of lines of code, yet the actual number of lines developed are not known until after the project is completed. Overall, the SLOC measurement technique was criticized throughout the literature. Most importantly, few people currently agree on what constitutes a line of code (5:22). This situation is attributed to a lack of well-defined rules for counting lines of code in each programming language (1:58; 21:13). For example, should we count only procedural lines of code or do we include declarative statements? How do we account for comments or blank lines of code? Do we count prototyped code or only delivered source lines of code?

Differing rules for counting SLOC is not the only factor contributing to sizing problems. There is also a lack of agreement for conversion factors when comparing code written in different languages. For example, high order languages are considered more efficient from a programming viewpoint than Assembly because one line of HOL code generates several lines of Assembly code. Therefore, when comparing a program written in Assembly code and another written in Ada code, an estimator should use a conversion factor to ensure an "apples-to-apples" comparison. One list of conversion factors claims

one line of COBOL is equivalent to three lines of Assembly, one line of PL1 equals 4.5 lines of Assembly, and 1 line of Smalltalk equals 15 lines of Assembly (21:14). However, there is currently no universally accepted convention for these equivalencies (21:14).

Due to the criticisms of using SLOC when measuring project size, several alternative techniques have been suggested. Function points, introduced by Allan Albrecht, are computed from five attributes of a program: the number of external input types, external output types, logical internal types, external interfaces types and external inquiry types (22:1-2). Several variations on Albrecht's function points have evolved including Adjusted Function Points, Mark II Function Points, and Capers Jones' Feature Points. Although some independent studies have verified function points are superior to SLOC for estimating cost and effort for business-related projects (22:2-3), a recent study indicates function points are less useful for estimating real-time projects (4:45).

Changing Requirements and Specifications. According to one software developer, the hardest part of building a software system is deciding exactly what to build. No other step in the development process cripples the resulting system more than a poorly defined system and no other part is more difficult to later rectify (17:17). As projects become larger and more complex, the importance of well-defined specifications cannot be understated. However, translating customer needs to written documents understandable by all parties continues to be a very difficult endeavor.

In his book, The Software Development Project: Planning and Managing, Phillip Bruce identifies well-defined software requirements as the cornerstone of a well-defined, well-understood, and well-costed software development effort (23:17). Vague, poorly written requirements hamper pricing and design efforts

by the developing contractor. Many times, program deficiencies resulting from poor requirements are not identified until the acceptance demonstration when the customer realizes the software does not have certain displays, interactive menus, or other desired features (23:17).

This does not suggest good requirements are carved in stone and never change. Software requirements are subject to change throughout the development process. Many factors contribute to these changes: the customer desires a more functional software package, new technologies are developed, or funding and mission priorities change. According to Alan Davis, user needs are constantly evolving; therefore, the system under development is always aiming at a moving target (24:1454).

However, the very nature of the software itself is often blamed for this changeability. Paul Rook made the following observations about software characteristics in a Software Engineering Journal article:

(1) Software has no physical appearance.

(2) It is deceptively easy to introduce changes into software.

(3) Effects of software changes propagate explosively. (25:7)

Normalization Explained

When asked how much they are willing to pay for a car, most consumers will adjust their offer price based on the presence (or absence) of various features such as air-conditioning, compact disk player, or other interior upgrades. This process of making adjustments for known differences is commonly referred to as normalization. The concept of normalization is

particularly important when comparing cost estimates generated by different cost models.

According to The AFSC Cost Estimating Handbook, proper use of any cost model requires a thorough analysis of the model's assumptions and limitations as well as its capabilities and features (9:8-6). This handbook points out several key questions the cost estimator should resolve before using any model to prepare an estimate. Specifically:

(1) What phases of the acquisition cycle are included in the model?

(2) Is the data required to use the model available?

(3) What units of measure are used by the model (dollars or manmonths)?

(4) What is the content of the data base on which the model was derived?

(5) What is the range of input values for which the model is valid? (9:8-6 to 8-10)

Unless the estimator understands the underlying assumptions of the model, which phases and activity costs are included in the estimate, and other such questions, it is impossible to fairly compare one model's cost estimate to a different model's cost estimate.

Cost Model Descriptions

REVIC. Revised Enhanced Version of Intermediate COCOMO is a non-proprietary cost model currently managed by the Air Force Cost Analysis Agency. Raymond L. Kile developed REVIC in response to an Air Force Contract Management Division initiative to improve oversight of government

software contract and for better support of software contract proposal evaluations (20). REVIC is a variation of the intermediate COCOMO described in Dr. Boehm's book Software Engineering Economics (1). According to the REVIC user manual, the model estimates development life cycle costs for software development from requirements analysis through completion of the software acceptance testing and a fifteen year maintenance life cycle (20:1). All equations used by the REVIC model are published in the REVIC user's guide.

The only required input necessary for running REVIC is a SLOC estimate for the proposed project (20:4). However, the user can tailor an estimate by adjusting various categorical variables used by REVIC. The model has four primary input categories: personnel, computer, product, and project attributes with 19 subordinate attributes. See Table 2-2 for REVIC inputs.

Table 2-2. REVIC Parameters by Category (20)

Personnel Attributes		Product Attributes	
Analyst Capability	ACAP	Requirements Volatility (Note 1)	RVOL
Programmer Capability	PCAP	Required Reliability	RELY
Applications Experience	AEXP	Data Base Size	DATA
Virtual Machine Experience	VEXP	Complexity	CPLX
Language Experience	LEXP	Required Reuse (Note 1)	RUSE
Project Attributes		**Computer Attributes**	
Modern Programming Practices	MODP	Time Constraints	TIME
Development Tools	TOOL	Storage Limitations	STOR
Project Security (Note 1)	SECU	Virtual Machine Volatility	VIRT
Development Schedule (Note 2)	SCED	Computer Turnaround Time	TURN
Management Reserve (Note 1)	MRES		

Note 1: Additional parameters not found in COCOMO.
Note 2: REVIC accepts only the nominal (1.00) value for SCED.

The user selects from available ratings for each parameter which range from Very Low (VL) to Extremely High (XX). Each rating translates to a

numerical value and when all 19 values are multiplied together, an environmental factor is calculated. This environmental factor is then used in REVIC's equations to calculate the project's schedule and cost.

The REVIC model differs from COCOMO model in several ways. The primary difference between REVIC and COCOMO is the basic coefficients used in the model's equations (20:4). REVIC was calibrated exclusively on DoD software development projects whereas COCOMO was not (13:5; 1:496). As a result, different coefficients were calculated when regression techniques were applied to the new set of calibration projects.

SASET. The Software Architecture Sizing and Estimation Tool is a non-proprietary model developed by Martin Marietta Corporation for the Naval Center for Cost Analysis (6:2-9). The Air Force Cost Analysis Agency is currently the central Air Force focal point and sponsor for this model (7). SASET is a forward-chaining, rule-based expert system utilizing a hierarchically structured knowledge database to provide functional software sizing values and an optimal software development schedule (26:1-2). Optimal development schedule refers to a derived schedule based on historical data reflecting the minimum development cost (26:1-3).

According to the SASET on-line tutorial, SASET uses a three-tiered approach to obtain input regarding environmental, sizing, and complexity factors related to proposed system (12). These three tiers are accompanied by fourth and fifth tiers which gather maintenance and risk analysis information. Tier 1 gathers information regarding type and class of software, programming language, number of CSCIs, and other development environmental factors. Tier 2 develops lines of code estimates based on user inputs regarding software functions, CSCIs, and CSCs.

SASET's sizing methodology is unique among the four models reviewed in this thesis. (New versions of SEER-SEM released since the beginning of this research effort also address function point sizing). Most cost models require the user to input the number of estimated lines of code; however, SASET allows the user to enter functional elements of the proposed system without any knowledge of the estimated number of lines of code. SASET will generate a line of code estimate based on the functionality specified by the user (12). The user also has the option of directly inputting an independent size estimate.

Tier 3 describes the software complexity issues of the proposed system. The user rates 20 complexity issues on a scale of 1 (very complex) to 4 (simple) (26:5-52). Examples of the complexity issues include system requirements, software documentation, development facilities, hardware constraints, software interfaces, and software experience (26:5-52 to 5-55).

Tiers 4 and 5 are not necessary for development effort estimation. Tier 4 defines the maintenance complexities and a maintenance schedule (12). Tier 5 provides risk analysis on sizing, schedule, and budget data. SASET uses Monte Carlo simulation techniques to create distribution curves and various tables of risk assessment parameters (26:1-3).

PRICE-S. PRICE-S is a commercially available software cost model distributed by PRICE Systems, a division of Martin Marietta Corporation. PRICE-S operates in a Microsoft Windows environment and features pull-down enus and an extensive on-line help facility. Due to its proprietary nature, all equations used by the model are not published; however, the PRICE-S Reference Manual and other reports describe essential equations and explain the methodologies used to develop the model (27, 28).

PRICE-S generates software cost estimates based on project scope, program composition, processor loading and demonstrated organization performance, and other environmental factors (27:A-1-1). PRICE-S inputs are grouped into seven main categories. (See Table 2-3 for model categories.)

Table 2-3. PRICE-S Input Variables (27:A-1-9)

1. Project Magnitude (How big?)
   The amount of code to be produced and the languages to be used.
2. Program Application (What Character?)
   The type of project such as MIS, Command and Control, Telemetry, Communications, etc.
3. Level of New Design and Code (How much new work is needed?)
   The amount of design and code that cannot be taken from existing inventory.
4. Productivity (Who will do the work?)
   The experience, skill, and know-how of the assigned individuals or team, as applicable to the specified task.
5. Utilization (What hardware constraints?)
   The extent of processor loading relative to its speed and memory capacity.
6. User Specifications and Reliability Requirements (Where and how used?)
   The level of requirements relating to testing, transportability and end use of the product.
7. Development Environment (What complicating factors exist?)
   The relative impact of unique project conditions on the normal time required to complete the job, measured with respect to the organization, resources, program application and project size.

Operational and testing requirements are incorporated, together with technology growth and inflation, to calculate values for six cost categories in nine development phases (PRICE-S considers one "pre-software" phase plus the eight standard development phases) (27:A-1-2). Cost categories calculated by PRICE-S include Design, Programming, Data, System Engineering/Program Management, Quality Assurance, and Configuration Management (27:A-1-2).

SEER-SEM. System Estimation and Evaluation of Resources - Software Estimation Model is a commercial cost model distributed by Galorath Associates, Incorporated and operates in a Microsoft Windows environment. SEER-SEM is based on a mathematical software estimation model developed by Dr. Randall W. Jensen (19:5-1). Although the original model was developed during 1979 - 1980, SEER-SEM makes use of the latest updates to its input parameters by Dr. Jensen, statistical conclusions from eight additional years of historical data, and the experience of software cost estimation experts (19:5-1). According to the SEER-SEM user manual, the model:

> (1) produces estimates for software developments from System Concept through 25 years of operational support
>
> (2) furnishes the estimator with a wide range of inputs to ensure proper representation of each unique software development
>
> (3) supplies basic software development estimation, design-to, and cost-to-complete capabilities
>
> (4) offers many, varied output reports to ensure the estimator has all the information necessary to estimate, monitor and control software development projects. (19:5-1)

SEER-SEM has four primary input categories including platform, application, development method, and development standard (19:Appendix E-1). The model uses these four inputs to select a set of default parameter inputs from integral knowledge bases. The only other required input is an estimated range for number of lines of code to be developed.

The user may fine-tune the model by altering the default input parameters selected from the knowledge base. Several key SEER-SEM parameters include complexity, personnel capabilities, development support environment, product

development requirements, product reusability requirements, development environment complexity, target environment, schedule, staffing, and software maintenance. The user must provide three values: one for the low value, one for the most likely value, and one for the high value (19:10-1).

The model also provides eighteen different reports analyzing cost, schedule, and input relationships and generates charts for staffing plans, schedule risk, and effort risk (19:11-1, 11-27). Some of the key reports generated by SEER-SEM include the Quick Estimate Report, Basic Estimate Report, Maintenance/Operation Support Report, and Inputs Report. The user can display these various reports in one window and the input parameters in another window. This allows the user to analyze how the cost estimate is impacted by changes in various input parameters.

Summary

This chapter reviewed recent publications and research efforts in the field of software cost estimation. Several factors impacting the accuracy of cost estimates were examined, the concept of normalization was explained, and model descriptions were obtained. The literature was consistent in one respect: software cost models, in and of themselves, do not automatically generate good cost estimates. One model developer best summed up this sentiment by stating:

> Estimation is not a mechanical process. Art, skill, and knowledge are involved .... The model's responsibilities are to transform measures of estimators into consistent projections for costs and schedules. The model cannot make the initial judgments that define the product, nor can it commit an organization to achieving goals. It .... does not replace estimators. It merely helps them do their job more quickly, more consistently, and more thoroughly. (28:61)

III. Methodology

Overview

This chapter discusses the methodology used for this research effort. Research was conducted in two phases: (1) independent analysis of the cost models and (2) validation and concurrence of analysis by cost model vendors and model experts.

Phase 1. Independent Analysis

The purpose of this phase was threefold. First, the researchers had to become familiar with each software model before any meaningful analysis could be performed. Second, after gaining familiarity with the models, the researchers examined the underlying assumptions and equations of each model to identify significant differences. Lastly, a baseline case study was developed and sensitivity analyses were conducted to determine the impact of identified differences between models.

During the first part of this phase, the researchers worked primarily on becoming familiar with each model under review. To achieve this objective, a review of current software cost estimating literature was conducted, previous model analyses were reviewed, and REVIC, SASET, PRICE-S and SEER-SEM software and user manuals were obtained. On-line tutorials for PRICE-S, SASET, and SEER-SEM were completed by the researchers (REVIC does not have this capability). Lastly, focal points for each model were identified to provide additional assistance as required. Telephone interviews were the primary means of communication with the focal points.

During the second part of this phase, the researchers examined the underlying assumptions and equations of each model to identify significant differences. To achieve this objective, the researchers relied heavily on the models' user manuals, technical reports, published articles, telephone interviews with model vendors, and hands-on experience with the models. To limit the risks associated with concurrent research (e.g. two team members conduct research on different models), a checklist was developed for examining the models. A brief excerpt of the checklist is provided below. See Appendix A for complete checklist used by researchers. Each issue was addressed as a specific finding in Chapter IV, Findings.

> Issue 1. What DoD-STD-2167A phases and activities are included in the estimate produced by each model?
>
> Issue 2. What general development activities and specific cost elements are estimated?
>
> Issue 3. How does each model define a source line of code and how are language differences accounted for?
>
> Issue 4. Which model factors are key cost drivers? What estimating methodology was used to develop these factors?
>
> Issue 5. How does the model account for schedule compression and extensions?
>
> Issue 6. What implications does project size have on model output?
>
> Issue 7. Are there any distinctive characteristics about the model's data base(s)?

During the third portion of this phase, a baseline case study was developed and sensitivity analyses were conducted to examine the impact of identified differences between models. The case study was based on a fictional

flight avionics system. The fictional system was programmed in Ada and developed using the traditional waterfall software development life cycle. See Chapter IV, Findings for additional details regarding the case study.

A fictional case study was preferred over actual program data for several reasons. Most importantly, it was the most flexible approach and allowed us to develop a workable case study within this research effort's time frame. Secondly, fictional data alleviated any potential problems with collecting and using proprietary data. Lastly, the purpose of this effort was to examine how the models estimate rather than to second-guess the cost estimates of previously completed projects or to compare the model's relative accuracies.

Using the information provided in the case study, cost estimates were generated using the nominal (default) values for all inputs in each model to determine the base cost estimates. With the assistance of the model developers/vendors, the researchers then adjusted model inputs to more accurately reflect the development environment for the fictional case.

Phase 2. Validation and Concurrence

This phase focused primarily on querying model vendors and model experts to obtain any additional information not found during independent research and validating research results with vendors. Important steps included:

1) Interviewing vendors/model experts to obtain necessary information not found in independent analysis.

2) Validating accuracy of research results with vendors/model experts.

3) Documenting results of validations/concurrences with vendors.

4) Reevaluating and updating, if necessary, research results after discussions with vendors.

This phase relied extensively on telephone and personal interviews with model vendors and experts. Additionally, there was an inherent degree of overlap between Phase 1 and Phase 2. Frequent conversations were necessary with model vendors since all equations and internal logic for the models are not published and readily available.

IV. Findings

FINDING #1. Software Development Phases

Software development efforts within the Air Force are guided primarily by two standards: DoD-STD-2167A for weapon system software and DoD-STD-7935A for management information systems. DoD-STD-7935A will not be discussed since this effort focused specifically on software development efforts related to new or modified weapon systems. DoD-STD-2167A establishes uniform requirements for software development efforts which are applicable throughout the weapon system's life cycle (10:B-7). (See Table 4-1 for DoD-STD-2167A phases.) Although the standard is not intended to specify or discourage use of any particular software development method, it outlines several major activities and key formal reviews the software development process should encompass.

Table 4-1. DoD-STD-2167A Phases/Activities and Key Formal Reviews or Audits (10:B-7)

Phase	Activity	Key Formal Reviews or Audits
Phase 1	System Requirements Analysis/Design	System Requirements Review (SRR) System Design Review (SDR)
Phase 2	Software Requirements Analysis	Software Specification Review (SSR)
Phase 3	Preliminary Design	Preliminary Design Review (PDR)
Phase 4	Detailed Design	Critical Design Review (CDR)
Phase 5	Coding and CSU Testing	None
Phase 6	CSC Integration and Testing	Test Readiness Review (TRR)
Phase 7	CSCI Testing	Functional Configuration Audit (FCA) Physical Configuration Audit (PCA)
Phase 8	System Integration and Testing	Formal Qualification Review (FQR)

Given the eight software development phases identified by DoD-STD-2167A, each software cost estimating model was examined to determine which phases were included in the estimate produced by each model and how effort was allocated among the development phases. Additionally, the model's ability to account for alternative approaches for implementing the software life cycle was reviewed. Table 4-2 summarizes the phases encompassed by the models and is followed by a detailed discussion of each model.

Table 4-2. Correlation Matrix of Software Development Phases

DoD-STD-2167A	REVIC	SASET	PRICE-S	SEER-SEM
System Rqmts Analysis/Design	**Not Addressed By Model**	System Concept System S/W Rqmts Analysis	System Concept System S/W Rqmts Analysis	S/W Rqmts Analysis **NOTE 1**
S/W Rqmts Analysis	S/W Rqmts Engineering	S/W Rqmts Analysis	S/W Rqmts Analysis	
Preliminary Design	Preliminary Design	Preliminary Design	Preliminary Design	Preliminary Design
Detailed Design	Critical Design	Detailed Design	Detailed Design	Detailed Design
Code & CSU Test	Code and Unit Testing	Code Unit Testing	Code and Test	Code & CSU Test
CSC Integration & Testing		CSC Informal Testing		CSC Integration & Test
CSCI Testing	Integration & Test	CSCI Formal Testing	CSCI Testing	CSCI Testing
System Integration & Test	Development Test & Evaluation	System Integration & Testing	System Test Operational Test & Evaluation	System Integrate Through Operational Test & Evaluation

Note 1: Initial research was performed with SEER-SEM, ver. 3.00. The current version (3.21) further allocates schedule/effort to System Requirements Analysis and S/W Requirements Analysis.

REVIC. REVIC estimates costs for six development phases versus the eight identified by DoD-STD-2167A phases. Although the model estimates effort associated with S/W Requirements Analysis, REVIC does not account for System Requirements Analysis/Design effort. REVIC also combines phases 5 and 6 of DoD-STD-2167A into a Code and Unit Testing phase (20:3).

REVIC's terminology for its last two phases can cause some confusion when comparing these phases against those identified by DoD-STD-2167A. REVIC's Integration & Test phase is actually comparable to DoD-STD-2167A's CSCI Testing phase. The REVIC User Manual defines the Integration & Test phase as the integration of CSCs into CSCIs and the testing of the CSCIs against the test criteria developed in the program (20:3). This effort corresponds to DoD-STD-2167A's definition of the CSCI test phase. Additionally, REVIC's Development Test & Evaluation phase is similar to DoD-STD-2167A's System Integration & Test. This phase includes testing the weapon system to ensure the requirements of the system level specifications are met.

Methodology for Allocating Development Effort Among Phases. REVIC initially calculates total software development effort based on user inputs for SLOC and various environmental factors. Total development effort is then allocated to four phases based on preset percentages. (See Table 4-3 for REVIC's default allocation percentages.) The model then adds 12% of total development effort for S/W Requirements and 22% for Development Test and Evaluation (DT&E) to account for the remaining two phases. For example, if the effort associated with Preliminary Design, Critical Design, Code & Unit Testing, and Integration & Test equals 1000 manmonths, REVIC adds 120 manmonths for S/W Requirements Engineering and 220 manmonths for the DT&E phase. The final estimate will total 1340 manmonths. Users may change the

Table 4-3. Default Allocation Percentages for REVIC Development Phases

Development Phase	Allocation Percentage
Preliminary Design	23%
Critical Design	29%
Code & Unit Testing	22%
Integration & Test	26%

percentages associated with these two phases to better match the distribution of effort applicable to their particular organization.

Primary and Alternative Approaches to the Software Life Cycle. REVIC is based on the waterfall life cycle approach. The model does not allow the user to specify other life cycle approaches such as evolutionary, prototype, or incremental development. Although REVIC has several development mode options (Ada, Organic, Semi-detached, and Embedded), these modes describe the type of development project and not the software life cycle approach.

SASET. If the user selects the DoD-STD-2167A life cycle option, SASET estimates cost and schedule milestones for ten phases versus the eight phases identified by DoD-STD-2167A. SASET breaks Phase 1 into two phases: a System Concept phase and a System Software Requirements Analysis phase. SASET also divides Phase 5 into two distinct phases: a Code phase and a Unit Test phase. All other SASET phases are equivalent to the phases described by DoD-STD-2167A.

Methodology for Allocating Development Effort Among Phases. Like REVIC, SASET first calculates total development effort and then allocates a percentage of the effort to each of the ten phases. The user can change the

default allocation percentages; however, the allocation percentages must add up to 1.0. (See Table 4-4 for the SASET's default allocation percentages.)

Table 4-4. Default Allocation Percentages for SASET Development Phases

Development Phase	Allocation Percentage
System Concept	7.5%
System S/W Requirements Analysis	8.5%
S/W Requirements Analysis	9.0%
Preliminary Design	7.0%
Detailed Design	17.0%
Code	13.0%
Unit Testing	7.0%
CSC Informal Testing	9.0%
CSCI Formal Testing	7.0%
System Integration & Testing	15.0%

It should also be noted SASET adds an integration penalty when several CSCIs must be integrated. However, the integration penalty is simply added to the total effort and distributed then among all ten phases based on the allocation percentage. The user has the option of assigning higher allocation percentages to the applicable phases to more accurately capture the additional effort involved with CSCI integration.

Primary and Alternative Approaches to the Software Life Cycle. SASET is based on the waterfall life cycle approach. The model does not have specific options which allow the user to select alternative life cycle approaches such as evolutionary, prototype, or incremental development. However, if the user was

extremely proficient with adjusting SASET calibration files, alternative life cycle approaches could be modeled (29).

PRICE-S. PRICE-S estimates cost and schedule milestones for nine phases of software development versus the eight phases identified by DoD-STD-2167A. PRICE-S breaks the System Requirements Analysis/Design phase into two phases: a System Concept phase and a System Software Requirements Analysis phase. PRICE-S also combines phases five and six of DoD-STD-2167A into a Code and Test phase. With the exception of the Operational Test & Evaluation phase, all other phases are equivalent to the phases described by DoD-STD-2167A.

The Operational Test and Evaluation phase accounts for contractor support costs related to the operational effectiveness and suitability of the deployed system. According to a PRICE-S model developer, this phase should be included if the project is a fairly simple effort (i.e. one or two contractors involved). However, if the effort is more complex and involves several contractors performing tests at various locations, the analyst should exclude this phase and estimate these costs outside of the model (30).

Methodology for Allocating Development Effort Among Phases. PRICE-S differs from REVIC and SASET in that no preset allocation percentages are used to distribute effort to the various software development phases. The core of PRICE-S consists of three major development activities: design, implementation and testing. These activities comprise Preliminary Design, Detailed Design, Code and Unit Test and CSCI Test. Key costs for the three major development activities are estimated and then a "ripple effect" submodel is used to distribute the costs across the software development phases. Figure 4-1 illustrates the ripple effect submodel used by PRICE-S.

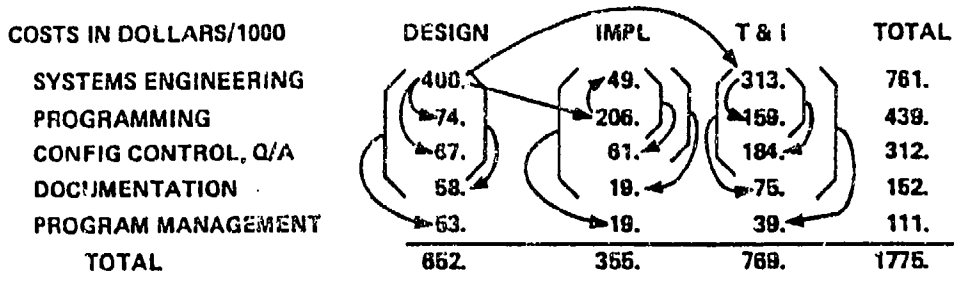


Figure 4-1. Ripple Effect for Calculating Activity Estimates (31)

The model developers viewed systems engineering during design as the key to software development effort (28:6). As systems engineering for a project increases, all other efforts necessary to complete the project will also increase. This is why the model developers call systems engineering "key" and use it as the starting point for the ripple effect.

According to "The Central Equations of the PRICE Software Cost Model", programming, configuration management, and quality assurance are viewed as supporting activities during the design phase (28:41). As a result, their effort and cost are driven by the amount of activity in systems engineering. This relationship is illustrated in Figure 4-1 by the two arrows immediately below the system engineering estimate. Documentation is a function of the three preceding activities and program management is a function of the sum of the four elements that need to be estimated.

Programming is the central activity in the implementation phase and the same approach used to estimate design costs is used to estimate supporting activities. During integration and testing, PRICE-S model developers elected to

use systems engineering as the driver and all supporting activities are calculated from this factor (28:41).

After these calculations are performed, the costs are rebalanced to account for several factors such as high utilization and specification levels. The Systems Engineering cost is apportioned to Design Engineering and System Engineering/ Program Management. Configuration/Quality Assurance is split into two cost elements: Configuration Management and Quality Assurance (28:41). The design activity is mapped to Preliminary Design and Detailed Design, implementation is mapped to Code and Unit Test, and integration and test is mapped to CSCI Test. Profiles are then created for each of the cost categories and used to distribute the costs over the development period.

Primary and Alternative Approaches to the Software Life Cycle. PRICE-S is based on the waterfall life cycle approach. The model does not have specific options which allow the user to select alternative life cycle approaches such as evolutionary, prototype, or incremental development. However, according to PRICE-S personnel, alternative life cycle approaches can be modeled (30). For example, an incremental life cycle approach can be modeled by setting up multiple CSCIs and adjusting global factors to represent the project's development profile. Inexperienced model users should seek assistance when attempting this process to ensure the desired results are achieved.

SEER-SEM. SEER-SEM estimates costs for nine software development phases. Overall, the phases closely parallel those identified by DoD-STD-2167A phases. Initial research was performed using SEER-SEM, version 3.00. Although this version of SEER-SEM estimates System Requirements Analysis effort; it is reported under the S/W Requirements Analysis phase and not specifically allocated to earlier phases (32). The current version (SEER-SEM,

ver. 3.21) addresses this issue and further allocates System Requirements Analysis effort to its associated phase.

Methodology for Allocating Development Effort Among Phases. According to the model developer, SEER-SEM allocates a percentage of the development schedule rather than effort to each phase (33). The allocation percentages are a function of the "Platform" and "Development Methods" knowledge bases. These knowledge bases specify the "Percent of Base Full Scale Implementation" allocated to various development phases. According to the SEER-SEM User's Manual, Full Scale Implementation (FSI) includes Preliminary Design, Detailed Design, Code and CSU Test, CSC Integration and Test (19:5-10).

The "Platform" and "Development Method" knowledge bases were examined to identify the default allocation percentages. (See Tables 4-5 and 4-6 for default allocation percentages.) Users may review and change, if desired, the allocation percentages by adjusting the values in the knowledge bases.

Table 4-5. Default Allocation Percentages for SEER-SEM Platform Knowledge Base

	Avionics	Business	Ground	Manned	Missile	Mobile	Others	Shipboard	Unmanned
% of Base FSI to PDR	20	20	20	20	20	20	20	20	20
% of Base FSI to CDR	43	50	43	43	43	43	43	43	43
% of Base FSI to CUT	70	70	70	70	70	70	70	70	70
% of Base FSI to TRR	97	97	97	97	97	97	97	97	97
% of Base FSI to FQT	100	100	100	100	100	100	100	100	100

Table 4-6. Default Allocation Percentages for SEER-SEM Development Method Knowledge Base

	AdaDev	AdaTrdl	AdaInc	Evolve	Incrmnt	None	Prototype	Spiral	Waterfall
% of Base FSI to PDR	0	20	20	13	20	1	1	23	0
% of Base FSI to CDR	0	45	45	35	45	3	3	42	0
% of Base FSI to CUT	0	75	75	95	75	97	95	95	0
% of Base FSI to TRR	0	97	97	97	97	98	97	97	0
% of Base FSI to FQT	0	100	100	100	100	100	100	100	0

An empirical analysis of SEER-SEM indicates the allocation percentages associated with non-FSI phases (Requirements Analysis, CSCI test, and System Integration and Testing) are generally independent of development method. Although the model developer states users "will see vastly different effort in requirement analysis depending on which knowledge bases are chosen", our analysis indicates the cost associated with requirements analysis was generally 5.8 percent of the total CSCI cost (33). (See Table 4-7 for an example of allocation percentages for a ground based radar system.) The reader should note these are only approximate allocation percentages for efforts involving the design, code, and test of 100% new code based on different development methods.

Primary and Alternative Approaches to the Software Life Cycle. SEER-SEM is unique among the models reviewed since specific options for alternative life cycle approaches are available The user chooses from eight development methods when describing the project and its CSCIs. (Note: Version 3.21 addresses a ninth development method: the evolutionary life cycle approach). Table 4-8 provides descriptions of each development method.

Table 4-7. Allocation Percentages for SEER-SEM Development Phases

Development Phase	Development Method							
	Ada Dev	Ada Full	Ada Increm	Traditional Increm	None	Prototype	Spiral	Waterfall
S/W Rqmts Analysis (Note 1)	5.8%	5.8%	5.8%	5.8%	5.8%	3.0%	5.8%	5.8%
Preliminary Design	9.0%	9.0%	9.0%	9.0%	0.3%	0.3%	47.5%	9.0%
Detailed Design	15.0%	16.5%	16.5%	16.5%	0.7%	0.6%	15.3%	15.0%
Code & CSU Test	23.5%	26.9%	27.0%	27.0%	74.5%	73.5%	10.7%	23.5%
CSC Integ. & Test	28.1%	23.1%	23.2%	23.2%	1.1%	2.3%	2.1%	28.1%
CSCI Test	3.2%	3.2%	3.2%	3.2%	2.2%	3.5%	3.2%	3.2%
System Integ. Thru OT&E	15.4%	15.4%	15.4%	15.4%	15.4%	16.7%	15.4%	15.4%

Note 1: This phase includes effort associated with both System Requirements Analysis and Software Requirements Analysis. Subsequently analysis of SEER-SEM, ver. 3.21 indicates the total effort allocated to System Requirements Analysis and Software Requirements Analysis remains at approximately 5.8%; however, the effort is now allocated to each phase rather than combined in the S/W Requirements Analysis phase.

Table 4-8. Descriptions of SEER-SEM's Development Methods (19:9-3 and 9-4)

Development Method	Description
Ada Development	Use of Ada as a programming language
Ada Development with Incremental Methods	Use of Ada as a programming language following the incremental development process
Ada Full Use	Use of Ada programming language, Ada development tools, and methods
None	No formal development process used
Prototype	Iterative development
Spiral	Cyclical model where a repeating set of activities are performed on an increasingly more detailed representation of the product
Traditional Incremental	Linear model which allows the developer to iterate among the activities within each of the life cycle phases for each of the increments defined for the system
Traditional Waterfall	Linear model where the activities of each phase of the life cycle must be completed before continuing to the next phase

SEER-SEM uses project and CSCI descriptions to select the appropriate knowledge base and its associated input values. (See Finding #7 for more information regarding SEER-SEM's knowledge bases.) Low level inputs are modified when different development methods are specified. For example, the "Adafull" option has different inputs for "Modern Development Practices Use" and "Language Type" to account for language differences and increased use of automated software tools. Likewise, inputs for "Requirements Volatility" are different when the "Incremental" option is selected. As a result, estimates vary when different development methods are specified.

Finding #2. Development Activities and Cost Elements

When comparing the final cost estimates produced by different models, it is important to recognize which development activities are (and are not) included in the model's output. While no regulatory requirement for a specific cost element format was identified, a framework of general development activities is suggested. Using this framework as a baseline, analysts can then identify and adjust for differences between the models.

Each model was examined to determine which general development activities are included in estimates. Table 4-9 provides a summary of the comparison between the baselined development activities to the activities estimated by each model. Table 4-10 identifies the specific cost elements estimated by each model and is followed by model definitions for each element.

Table 4-9. General Development Activities included in Model Estimates

	REVIC	SASET	PRICE-S	SEER-SEM
Rqmts Effort	Note 1	Yes	Yes	Yes
Design	Yes	Yes	Yes	Yes
Code	Yes	Yes	Yes	Yes
Test Activities	Yes	Yes	Yes	Yes
Quality Assurance	Yes	Yes	Yes	Yes
Configuration Mgt	Yes	Yes	Yes	Yes
Direct S/W Program Mgt	Yes	Yes	Yes	Yes
Documentation	Note 2	Yes	Yes	Yes

Note 1: System Requirements Analysis not addressed by REVIC.

Note 2: Does not reflect all effort necessary to conform to DoD-STD-2167A documentation requirements.

Table 4-10. Specific Cost Elements/Activities Estimated by Each Model

Model	Cost Element/Activity
REVIC	• Requirements Analysis • Product Design • Programming • Test Planning • Verification and Validation • Project Office Functions • Configuration Management and Quality Assurance • Manuals
SASET	• Systems Engineering • Software Engineering • Test Engineering • Quality Engineering
PRICE-S	• Design • Programming • System Engineering/Project Management • Configuration Control • Quality • Data
SEER-SEM	• Requirements Analysts • Software Design • Software Programming • Software Test • Project Management • Configuration Management • Quality Assurance • Data Preparation

REVIC. REVIC distributes effort between eight major cost elements or project activities. Definitions and activities associated with REVIC's cost elements are listed in Table 4-11.

Table 4-11. REVIC Cost Elements and Definitions (1:49)

Requirements Analysis. Determination, specification, review and update of software functional, performance, interface, and verification requirements.
Product Design. Determination, specification, review and update of hardware-software architecture, program design, and data base design.
Programming. Detailed design, code, unit test, and integration of individual computer program components. Includes programming, personnel planning, tool acquisition, data base development, component level documentation, and intermediate level programming management.
Test Planning. Specification, review and update of product test and acceptance test plans. Acquisition of associated test drivers, test tools, and test data.
Verification and Validation (V&V). Performance of independent requirements validation, design V & V, product test, and acceptance test. Acquisition of requirements and design V & V tools.
Project Office Functions. Project level management functions. Includes project level planning and control, contract and subcontract management, and customer interface.
Configuration Management and Quality Assurance. Configuration management includes product identification, change control, status accounting, operation of program support library, development and monitoring of end item acceptance plan. Quality assurance includes development and monitoring of project standards, and technical audits of software products and processes.
Manuals. Development and update of user, operator, and maintenance manuals.

The amount of effort associated with each activity is a percentage of the effort allocated to the following development phases: Software Requirements Engineering, Preliminary Design, Programming (including Critical Design and Code & Test), and Integration & Test. For example, 46% of the effort incurred in the Software Requirements Engineering phase is allocated to the requirements

analysis activity. The effort required in Development Test & Evaluation phase is distributed among the eight activities. The effort distributed to each activity is reported in REVIC's Activity Distribution Report and summarized in Table 4-12. The percentage allocated by the Software Requirements Engineering phase changes slightly for projects larger than 128,000 SLOC. Specifically, 44% of the effort (versus 46%) is allocated to requirements analysis.

Table 4-12. Default Allocation Percentages for REVIC's Activities

Activity	Development Phase			
	S/W Rqmts Eng	Preliminary Design	Programming	I&T
Rqmts Analysis	46%	10%	3%	2%
Product Design	14%	42%	6%	4%
Programming	6%	12%	55%	40%
Test Planning	4%	6%	6%	4%
Verification & Validation	8%	8%	10%	25%
Project Office Functions	12%	11%	7%	8%
CM/QA	4%	3%	7%	9%
Manuals	6%	8%	6%	8%

Project Management and Documentation Costs. REVIC includes the cost of project management and documentation in its effort calculations. Project management costs are limited to direct project management and project office functions. REVIC does not estimate indirect higher management costs (1:59).

More importantly, REVIC does not estimate all data requirements necessary to conform to DoD-STD-2167A. According to the REVIC model developer, less than 15% of the estimate is attributed to the cost of

documentation (34). REVIC implicitly includes the cost of a Requirements Analysis document, a Detailed Design document, and Test documentation in its estimate. However, the effort required to generate many of the documents required by DoD-Std-2167A is not included in the estimate. As a result, REVIC will tend to underestimate documentation costs for projects based on DoD-STD-2167A. Users can refer to REVIC's CDRL section to aid in calculating additional effort necessary to meet DoD-STD-2167A documentation requirements.

SASET. SASET allocates effort among four engineering organizations rather than traditional cost elements. These organizations are Systems Engineering, Software Engineering, Test Engineering, and Quality Engineering. The primary activities associated with each organization are listed in Table 4-13.

Table 4-13. SASET Cost Elements and Definitions (29)

Systems Engineering. Involves analysts who translate and analyze customer requirements, develop requirements and testing parameters, and serve as a customer interface.
Software Engineering Utilizes requirement information obtained from Systems Engineering and derives requirements, applies requirements to S/W design and architecture, and implements design (coding).
Test Engineering. Responsible for developing the test plan and test procedures, executing tests, and reporting results.
Quality Engineering. Oversees product development and ensures the development plan is carried out and properly executed. Verifies requirements are accomplished, customer expectations are satisfied, and the product is produced in accordance with company practices.

SASET allocates 14% of the total effort to systems engineering, 66% of the total effort to software engineering, 15% to test engineering, and 5% to quality engineering; however, these percentages can be changed by the user. SASET then further allocates organizational effort in each phase of the software

life cycle. For example, based on SASET's default calibration settings, 9.6% of the effort required for the code phase is allocated to Systems Engineering, 72.9% to Software Engineering, 12% to Test Engineering, and 5.5% to Quality Engineering. Users can refer to SASET's "Summary of Software Development Effort by Organization & Phase Report" to determine how the effort is allocated among the various development phases.

Program Management and Documentation Costs. SASET calculates direct software program management costs and documentation costs; however, neither are specifically broken out. SASET does not specifically break out documentation costs for software development projects. According to Martin Marietta personnel, the model does account for documentation costs and that approximately 22% of the total development effort is attributed to documentation effort and is distributed throughout the project's life cycle (29).

PRICE-S. PRICE-S estimates six major cost elements for software development efforts. The cost elements include design, programming, systems engineering/project management, configuration management, quality, and data. Definitions for PRICE-S cost elements and definitions are listed in Table 4-14.

Program Management and Documentation Costs. PRICE-S specifically accounts for all costs associated with software documentation and direct software program management. Documentation costs are accounted and reported within the DATA cost element. Although program management is not specifically broken out, it is accounted and reported within the "System Engineering/Project Management" cost element. This cost element includes the cost of System Engineering activities as well as Project Management.

Table 4-14. PRICE-S Cost Elements and Definitions (27:A-10 and A-11)

Design. The design cost element contains all costs attributed to the software engineering design department. These costs include engineering supervision, technical and administrative support and vendor liaison required for the software development effort.
Programming. The programming cost element contains all costs attributed to writing the source code and subsequently testing it.
System Engineering/Project Management. This element includes the System Engineering to define the software system, and the Project Management effort required to manage the software development project. The system engineering activity encompasses the effort to define the system requirements, and the integrated planning and control of the technical program efforts of the design engineering, specialty engineering, development of test procedures, and system oriented testing and problem resolution. Project Management efforts include managing the software development program in accordance with all procedures identified in the Software Development Plan, design review activities, and administrative duties.
Configuration Control. This activity involves the determination, at all times of precisely what is, and is not, an approved part of the system. To accomplish this, it is necessary to perform three tasks. The first involves incorporating requirements and specifications into the Functional and Allocated Baselines. Once a document has been incorporated into the baseline, changes may only be made through the configuration control task. This task involves the evaluation of changes and corrections to the baseline. Finally, it is necessary to provide for the dissemination and control of approved baseline material. Configuration Control also review the test procedures and ensure compliance with test plans and specifications.
Quality. This cost element includes the effort required to conduct internal reviews and walk-throughs to evaluate the quality of the software and associated documentation. Activities included in this element are evaluation of the Software Development Plan, software development library maintenance, and the Software Configuration Management Plan.
Data. This cost element contains all costs associated with software deliverable documentation. For military platforms, this includes responding to the "Contractor Data Requirements List" (CDRL) which contains requirements for delivery of all requirements, design, maintenance, and user manuals (i.e. Systems Segment Specification, Top Level Design and Detailed Design Specifications, Programmer and User Manuals, etc.).

According to a recent internal PRICE-S report, a general rule is that sixty percent of the total System Engineering/ Project Management costs are for Project Management and the remaining forty percent of costs are attributed to Systems Engineering (35). See Figure 4-2 for an example of a PRICE-S report and associated cost elements.

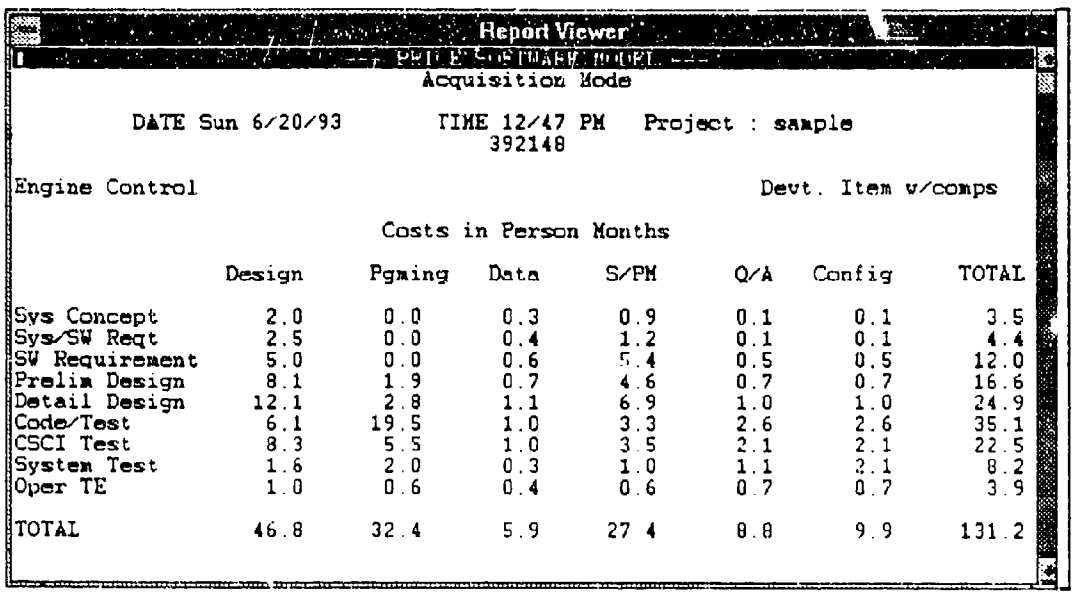


Figure 4-2. Example of PRICE-S Report with Cost Elements

SEER-SEM. SEER-SEM estimates eight major labor categories for software development efforts. Definitions and activities associated with each category are listed in Table 4-15.

Program Management and Documentation Costs. Although SEER-SEM, version 3.0 accounts for direct software program management and documentation costs, the specific costs allocated to each of these elements cannot be determined. However, newer versions (SEER-SEM, version 3.21) explicitly break out total effort among the various labor categories. See Figure 4-3 for an example of a SEER-SEM report and associated labor categories.

According to SEER Technologies personnel, all documentation costs are not captured by the Data Preparation category (36). For example, effort associated developing user manuals is associated with both the Data Preparation and Software Design categories. As a result, users should not report total documentation costs based solely on the Data Preparation category.

Table 4-15. SEER-SEM Cost Elements and Definitions (19:5-11 and 5-12, 36)

Requirements Analysts. Responsible for developing S/W requirements and specifications.
Configuration Management. CSCI configuration identification, change control, configuration status accounting, and configuration auditing to ensure proper configuration control.
Program Management. Direct labor management. It does not include hardware management, highest level program management, etc.
Quality Assurance. Includes the quality engineering functions (ensures quality is built into the product and developing appropriate standards), and quality control inspection and audits.
Software Design. The definition of the software architecture to implement the software requirements, preparation of architectural design specifications, design reviews, layout of physical data structures, interfaces, and other design details to implement the requirements.
Software Programming. The actual coding, unit testing, maintaining appropriate unit documentation, and test driver development for the individual software modules/CSUs.
Software Test. Preparing test plans and procedures, running tests, and preparing test reports. This includes software-related tests only.
Data Preparation. Effort to prepare specifications, standard engineering draft manuals (only includes engineering effort) and other engineering paper products.

...lfects\1 100K GSLOC Person Months by Labor Category

File Edit Estimate Input Report Chart Options Window Help

Activity	Management	Requirements	Design	Code
System Concept	0.00	0.00	0.00	0.00
System Requirements Design	2.55	9.77	2.97	1.27
S/W Requirements Analysis	7.48	28.67	8.73	3.74
Preliminary Design	14.15	12.86	52.73	15.43
Detailed Design	23.73	21.57	88.44	25.89
Code & CSU Test	23.64	10.13	28.27	185.77
CSC Integrate & Test	32.26	8.07	16.13	157.27
CSCI Test	3.71	0.93	1.85	18.00
System Integrate Thru OT&E	17.65	4.41	8.83	83.85
Maintenance / Op Support	0.00	0.00	0.00	0.00
TOTAL	125.17	96.42	199.95	491.30

Activity	Data Prep	Test	CM	QA
System Concept	0.00	0.00	0.00	0.00
System Requirements Design	1.27	2.55	0.42	0.42
S/W Requirements Analysis	3.74	7.48	1.25	1.25
Preliminary Design	10.29	10.01	2.57	2.57
Detailed Design	17.26	30.20	4.31	4.31
Code & CSU Test	20.27	50.66	13.51	13.51
CSC Integrate & Test	32.26	116.95	20.16	20.16
CSCI Test	3.71	13.45	2.32	2.32
System Integrate Thru OT&E	2.21	86.26	11.03	4.41
Maintenance / Op Support	0.00	0.00	0.00	0.00
TOTAL	91.00	327.55	55.58	48.96

Figure 4-3. Example of SEER-SEM, ver. 3.21 Report with Labor Categories

FINDING #3. Source Lines of Code and Language Differences

Lines of code is a commonly used metric for measuring the size of software development efforts. Yet, there are many different techniques for counting lines of code such as delivered source instructions, executable statements, terminal semi-colons, etc. Due to the numerous conventions for counting lines of code, it is important that analysts understand how specific models define a line of code. For example, if a model defines a line of code as all executable statements and comments, then the model user should ensure inputs for project size include comments as well as all executable statements.

Users should also be aware of how different programming languages impact development effort. For example, it may take five lines of code in one language to perform the same operation that requires 10 lines of code in another language. Figure 4-4 illustrates a sample program operation written in Ada and the same operation written in Assembly. The sample program operation computes: if "x" is less than four, then "y" becomes equal to seven; otherwise, "y" becomes equal to "x" plus five.

```
Ada :          if x < 4 then y := 7; else y := x+5; endif:

Assembly:      MOV x, Ax
               CMP Ax, 4
               JL label
               INC Ax, 5
               MOV Ax, y
               JMP end
          label MOV 7, y
          end
```

Figure 4-4. Sample Ada and Assembly Program Operation (12)

In this example, it takes eight lines of Assembly code to execute the same operation as one line of Ada. However, this does not necessarily mean it takes eight times as much effort to develop a program in Assembly than Ada since total development effort is a function of many other environmental factors.

REVIC. REVIC, a COCOMO variant, refers to lines of code as source instructions. COCOMO's developer defined source instructions as:

> All program instructions created by project personnel and processed into machine code by some combination of pre-processors, compilers, and assemblers. It excludes comment cards and unmodified utility software. It includes job control language, format statements, and data declarations. Instructions are defined as code or card images. Thus, a line containing two or more source statements counts as one instruction; a five-line data declaration counts as five instructions. (1:59)

When the effort involves adapted (or modified) code, REVIC adjusts SLOC using Equation 1.

Equation (1): $$EDSI = ADSI * \left[\frac{0.4DM + 0.3CM + 0.3IM}{100}\right]$$

where:

EDSI	=	Equivalent Delivered Source Instructions
ADSI	=	Adapted Delivered Source Instructions
DM	=	Design Modification
CM	=	Code Modification
IM	=	Retesting

In Equation 1, ADSI is multiplied by the percent of design modification, code modification, and retesting. For example, an adapted code package which had exactly 100% design modification, 100% code modification, and 100% retest would result with an EDSI equal to the ADSI (20:14).

Model's Ability to Account for Different Languages. REVIC does not differentiate between languages. The model calculates the same effort for 10,000 lines of Assembly as for 10,000 lines of FORTRAN. The only exception is if the Ada development mode is selected. Estimates based on the Ada development mode generally result in less effort than the embedded and semi-detached development modes, but more effort than the organic development mode. (See Finding #5 for REVIC's equations for different development modes.)

SASET. SASET defines SLOC as:

> All executable statements, plus inputs/outputs, format statements, data declaration statements, deliverable job control language statements, and procedure-oriented language statements. SLOC does not include statement continuations, database contents, "continue" statements, or program comments. (12)

Users may specify the amount of new, modified, and rehosted code associated with the development effort. New code is software code that will be developed from scratch. Modified code is software code which has some development already complete and can be used in the software program under consideration. Rehosted code is completed and tested software code which will be transferred from one computer system to the new system under development.

Once inputs for the SLOC values are made, SASET computes an adjusted sizing value called "New HOL Equivalent Source Lines of Code". To derive this value, SASET assumes that 73% of the modified code and 10% of the rehosted code is equivalent to the effort required to generate a new LOC. These percentages can be changed by modifying the SASET TIERS.CAL calibration file. For example, 1000 lines of modified code is equivalent to the effort required to generate 730 lines of new code. SASET also assumes every three lines of Assembly code is equivalent to one line of HOL code.

Table 4-16 provides an example of how SASET calculates the New HOL Equivalent SLOC.

Table 4-16. Sample SASET Calculations for New HOL Equivalent SLOC

New HOL	Modified HOL	Rehost HOL	New ASSY	Modified ASSY	Rehost ASSY	Total SLOC
1000	1000	1000	1000	1000	1000	6000
New HOL Equivalents:						
1000	730	100	333	243	33	2439

Calculations:

New HOL	=	(1000 x 1.00)	=	1000 lines of New HOL
Modified HOL	=	(1000 x 0.73)	=	730 lines of New HOL
Rehost HOL	=	(1000 x 0.10)	=	100 lines of New HOL
New ASSY	=	(1000 / 3)	=	333 lines of New HOL
Mod ASSY	=	(1000 / 3) x 0.73	=	243 lines of New HOL
Rehost ASSY	=	(1000 / 3) x 0.10	=	33 lines of New HOL

Model's Ability to Account for Different Languages. SASET allows the user to select from two categories of language types: high-order languages (HOLs) and Assembly language. The model does not differentiate between HOLs. For example, SASET considers a line of Ada equivalent to a line of CMS, FORTRAN, or any other commonly recognized HOL. The user may also specify the amount of new, modified, and rehosted code in HOL and/or Assembly language.

PRICE-S. According to the PRICE-S Reference Manual, source lines of code are defined as:

> The total number of s urce lines of code to be developed and/or purchased. Comments embedded in the code are not to be counted. However, type declarations and data statements should be included and will be broken out separately via the FRAC input. (27:37)

PRICE-S allows the user to specify the percentage of new design and new code. Application-weighted averages for new design and new code are used since some parts of a software system are more difficult to design and implement than others (27:D-1-35). This effect is illustrated by this example:

> If one were to estimate the costs for a system in which 50% of the code is to be reused, reasoning might lead to the conclusion that the effort required would be about half that for 100% new code. This reasoning would fail to recognize that it is invariably the inexpensive code (utilities, math, etc.) that is available for reuse, and that the difficult, machine dependent, problem-specific code has yet to be constructed. (27:D-1-35)

PRICE-S modifies the user input for SLOC using a composite application value (APPL). Equations 2 through 4 on the following page are used to compute the application-weighted factors and modified SLOC. Furthermore, the model assumes off-the-shelf code is not free since programmers must become familiar with the code, the final test and integration of subsystems will involve all new or modified software, and all delivered software must be documented (27:A-1-12).

Model's Ability to Account for Different Languages. PRICE-S allows the user to select from 20 different programming languages. If a specific development language is not listed, the user may choose from four generic groupings. Table 4-17 lists language selections within PRICE-S.

SEER-SEM. SEER-SEM defines SLOC as all executable source lines of code such as control, mathematical, conditional, deliverable Job Control Language (JCL) statements, data declaration statements, DATA TYPING and EQUIVALENCE statements, and INPUT/OUTPUT format statements. Source lines of code exclude comment statements, blank lines, BEGIN statements from Begin/End pairs, non-delivered Programmer Debug statements, continuation of format statements, machine/library generated data statements (19:10-2).

Equation (2): $NEWC = \frac{\sum_i (MIX_i * APPL_i * CODE_i)}{APPL}$

Equation (3): $NEWD = \frac{\sum_i (MIX_i * APPL_i * DESIGN_i)}{APPL}$

Equation (4): $SLOCM = SLOC * APPL$

where:

APPL	=	$APPL = \sum_i MIX_i * APPL$
MIX_i	=	Fraction of total SLOC devoted to performing functions in application category$_i$
$APPL_i$	=	Application value for functional application category
$CODE_i$	=	Fraction of code of application category$_i$ representing new work
$DESIGN_i$	=	Fraction of design of application category$_i$ representing new work
SLOCM	=	Modified Source Lines of Code

Table 4-17. Language Selections within PRICE-S (27:C-3)

1	Ada	9	COBOL	17	PASCAL
2	ALGOL	10	COMPASS	18	PL1
3	APL	11	CORAL66	19	PRIDE
4	ASSEMBLY	12	FLOD	20	SPL1
5	ATLAS	13	FORTRAN	21	HIGHER ORDER
6	BASIC	14	IFAM	22	MACHINE
7	C	15	JOVIAL	23	4th GENERATION
8	CMS	16	MICROCODE	24	INTERPRETIVE

SEER-SEM allows the user to allocate SLOC between three categories: new lines of code, pre-existing (not designed for reuse) code, and pre-existing (designed for reuse) code. Using these inputs, the model calculates an "Effective Size" which allows comparison of development alternatives when some alternatives include reusable code and others do not (19:1-1).

New lines of code are those lines that will completely designed, implemented, and tested. Pre-existing (not designed for reuse) code involves

lines that were not originally designed for reuse but will be used during this effort; whereas pre-existing (designed for reuse) code was specifically designed to ensure reusability. The user further allocates SLOC among six sub-categories: pre-existing lines of code, lines to be deleted in pre-existing code, lines to be changed in pre-existing code, percent to be redesigned, percent to be reimplemented, and percent to be retested.

Model's Ability to Account for Different Languages. SEER-SEM allows the user to select from a variety of programming languages. Table 4-18 lists the choices available for the Language Type (complexity) parameter. The model also accounts for differences associated with the Ada language with its Development Method knowledge base (AdaDev, AdaFull, and AdaInc).

Table 4-18. Language Selections within SEER-SEM (19:10-20 and 10-21)

Rating	Description
Very High	Full Ada, PI/I Version F
High	JOVIAL, CMS-2, Mainframe Assemblers
Nominal	Pascal, FORTRAN, COBOL, C, PL/1 Subset G, PC Basic, Micro Assemblers, Ada without Tasking
Low	Basic, Many 4GLs

Note:The user may select in between these setting with the PLUS and MINUS ratings. For example, nominal + is higher than nominal; nominal - is lower than nominal.

Finding #4. Key Model Attributes and Key Cost Drivers

Although different software cost models calculate total development effort in different ways, the model developers appear to agree in one respect. Project costs cannot generally be estimated solely on the basis of the number of source lines of code. A variety of factors such as programmer capabilities, modern development practices, requirements volatility, and reusability requirements influence the development effort.

Recognizing the need to account for these factors, the models allow users to describe the development environment by modifying various development attributes (or inputs). For example, if the user knows the development team has extensive programming experience for the proposed project, he or she may adjust a model input to reflect these circumstances. If no additional information is available, the user should select the nominal value.

This section reviewed model inputs and four broad categories were identified: personnel capabilities, development environment, target environment, and project requirements. Many of the key model inputs were categorized in Table 4-19; however, this table does not attempt to categorize all inputs nor does it illustrate equivalent model inputs. It is provided to show a sampling of available model inputs for the previously identified categories. Additionally, the reader should not assume the model inputs are equivalent simply because they are lined up across from one another.

The impact of various model inputs on total development effort was also examined. The results of this analysis were provided in the form of graphs and figures, where possible, to illustrate the subtle (and not so subtle) impact on model estimates.

Table 4-19. Categorization of Key Model Attributes

	REVIC	SASET	PRICE-S	SEER-SEM
Personnel Capabilities	ACAP PCAP AEXP VEXP LEXP	Development Team S/W Experience H/W Experience Integration Exp Personnel Resources	CPLX1 PROFAC	Analyst Capability Analyst Application Exp Programmer Capability Programmer Lang Exp Target System Exp
Development Environment	MODP TURN VIRT TOOL SECU	Dev Locations Workstation Types Programming Lang S/W Dev Tools Dev Facilities Travel Requirements	CPLX1 CPLX2 CPLXM PROFAC LANG SSR Date	Modern Dev Practices Use Resource Dedication Multiple Site Development Automated Tool Use Language Type Terminal Response Time
Target Environment	TIME STOR RELY DATA CPLX RUSE	% of Memory Utilized % of Microcode Timing & Criticality Man Interaction Hardware Constraints Software Interfaces	UTIL APPL NEWC NEWD PLTFM MIX	Memory Constraints Time Constraints Security Rqmts Target System Volatility Target System Experience Real Time Code
Project Rqmts	RVOL	System Rqmts S/W Rqmts	PROFAC CPLX1	Rqmts Volatility Specification Level

Note: This table does not attempt to illustrate equivalent model inputs. It is provided to show a sampling of available model inputs for four main development categories.

REVIC. REVIC initially calculates development effort as a function of SLOC and the development mode. After adding additional effort for the software requirements analysis phase and development test and evaluation phase, REVIC multiples total effort by an environmental modifier which consists of 18 model inputs.

The environmental modifier is the product of the values for each model attribute. As a result, REVIC's environmental modifier can have a significant impact on the final estimate. In a worst case scenario where input parameters are set to the most difficult setting, the environmental modifier will equal 450.703. In a best case scenario, the environmental modifier will equal 0.055. Thus, if the nominal effort for a project is 500 manmonths, the effort could range from 27.5 manmonths (best case scenario) to 225,351.5 manmonths

Due to the sensitive nature of certain REVIC inputs, Figures 4-5 through 4-8 are provided to show the range and impact of various REVIC inputs. The impact of schedule compression and extensions are not addressed in this section. (See Finding #6 for details regarding this issue.)

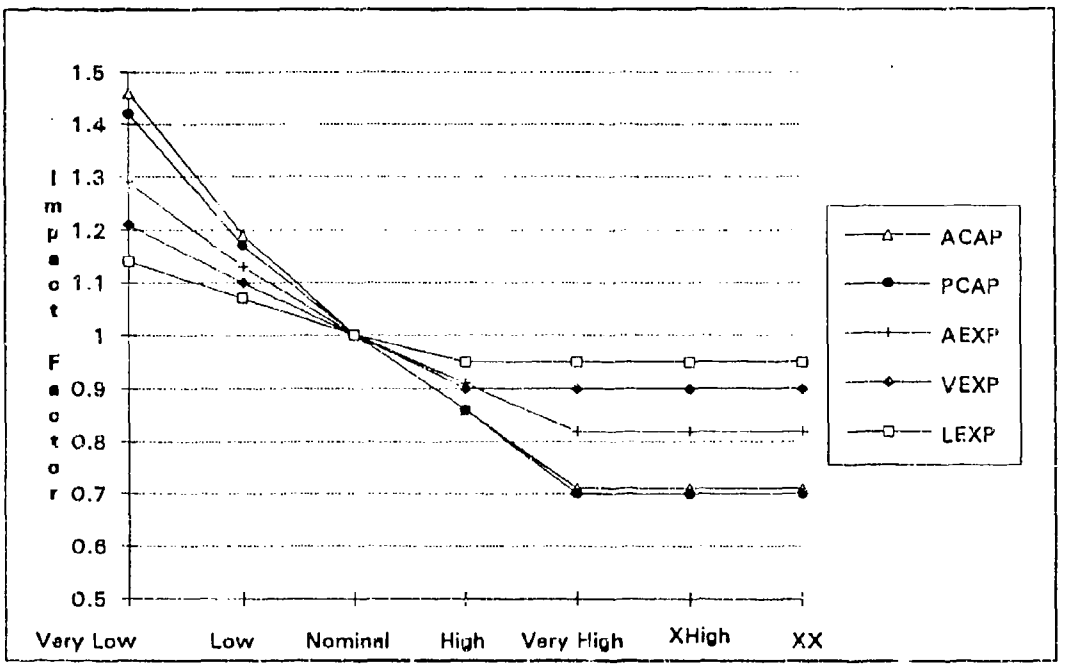


Figure 4-5. Impact of REVIC Inputs for Personnel Capabilities and Experience

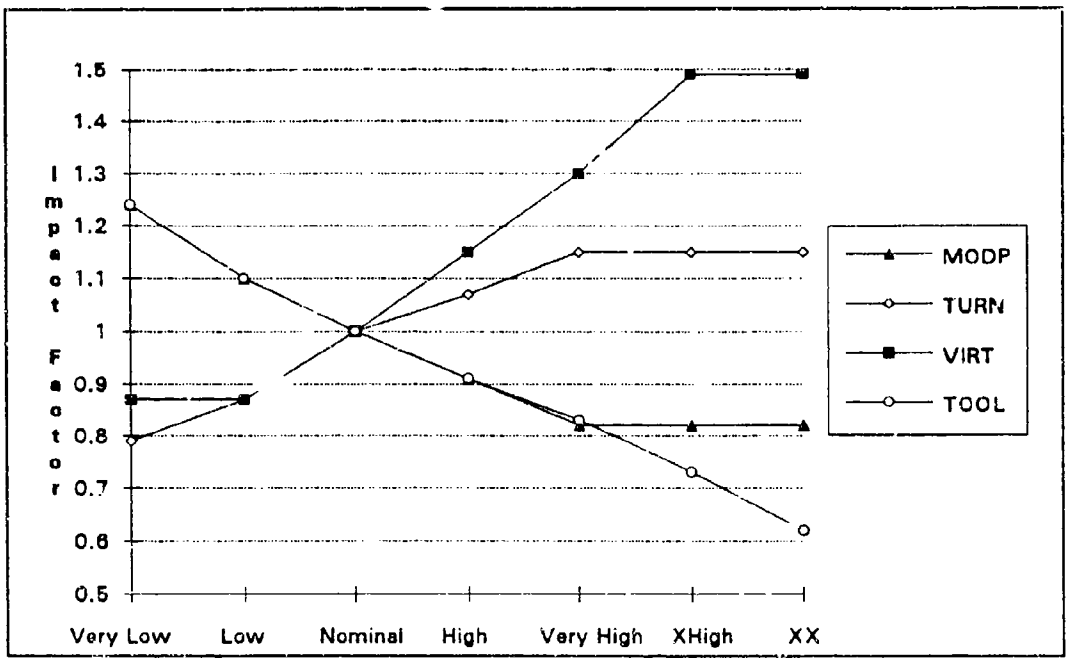


Figure 4-6. Impact of REVIC Inputs for Development Environment

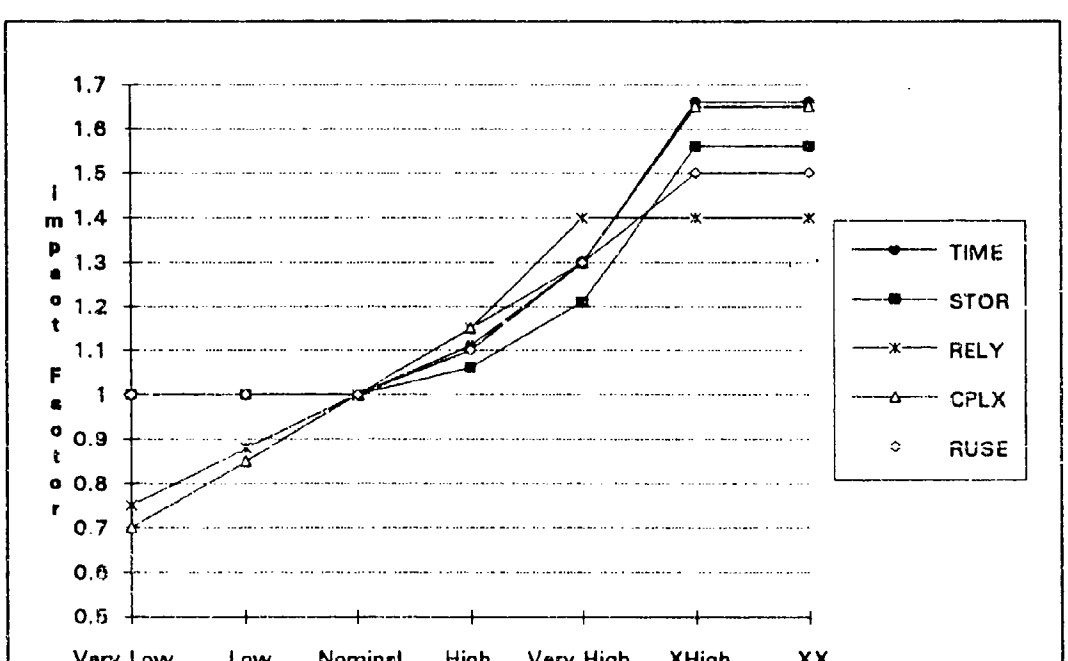


Figure 4-7. Impact of REVIC Inputs for Target Environment

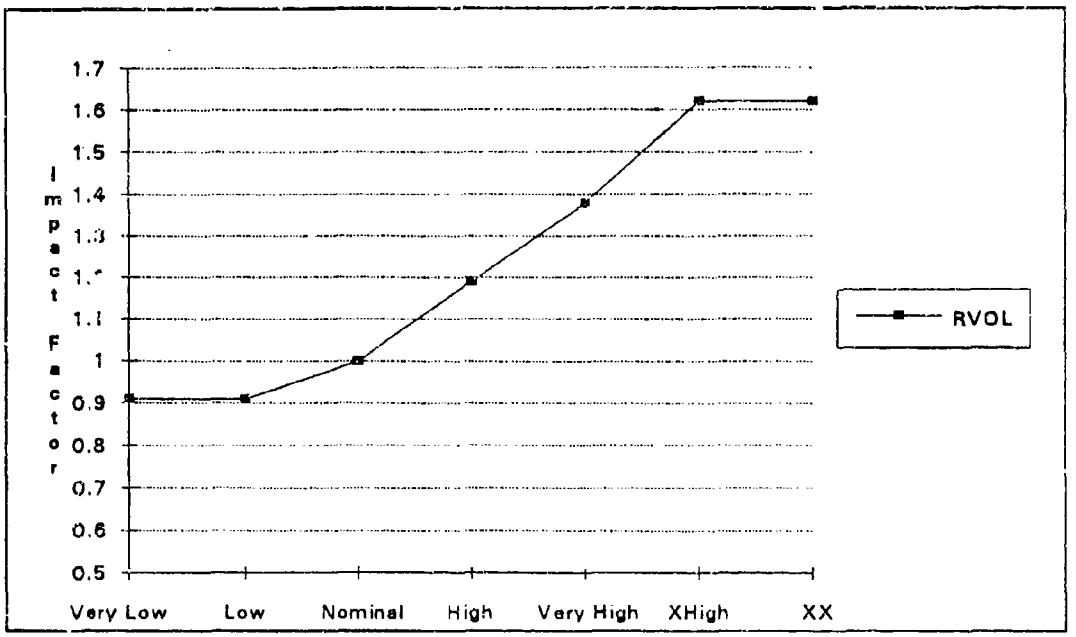


Figure 4-8. Impact of REVIC Inputs for Project Requirements

SASET. Development effort is a function of the lines of new equivalent HOL code, software type, and software class. The values for software type and class of software were developed by regression analysis on Martin Marietta's database of 500 projects (29). Tables 4-20 and 4-21 list the default values for various software types and classes.

The following example is provided to illustrate how the core of SASET's estimate is calculated. A manned, flight application program with 50,000 lines of equivalent HOL code would required 285,000 development hours (50,000 lines * 3.0 manhours/line * 1.9 manhours/line). This value represents the core of SASET's estimate; however, the model considers other environmental factors. According to Martin Marietta personnel, these factors should not be considered primary drivers, but they can influence the estimate (29). Consequently, users should understand how they impact the estimate.

Table 4-20. Default Values for SASET Software Types

Software Type	Default Values (Manhours/SLOC)
Systems	3.30
Applications	1.90
Support (Note 1)	0.85
Data Statements (Note 2)	0.075

Note 1: The SASET 3.0 Technical Reference Manual (Beta Version) recorded the support value as 1.1. However, SASET's calibration file has a value of 0.85. According to Martin Marietta personnel, 0.85 is the correct value (29).

Note 2: The SASET 3.0 Technical Reference Manual inaccurately states Data Statements are multiplied by Software Class value. The Data Statements value (.075) is actually multiplied by the corresponding Data Factors in the calibration file.

Table 4-21. Default Values for SASET Software Classes

Software Class	Default Values (Manhours/SLOC)
Manned Flight	3.00
Unmanned Flight	2.30
Avionics	1.80
Shipboard/Submarine	1.35
Ground	1.00
Commercial	0.70

The core estimate may be impacted by the Average Software Budget Multiplier (ASBM). The ASBM is the average of the Software Budget Multiplier (SBM) and Software System Budget Multiplier (SSBM). The SBM is calculated from the Tier 1 inputs addressing the system environment. The user may notice that software class is a Tier 1 input; however, it is not included in the SBM

calculation. The SSBM is calculated from the Tier 3 inputs addressing the attributes of the system and each CSCI has its own Tier 3 section. As a result, CSCIs from the same project may have the same SBM values but different SSBM values.

The ASBM is analogous to the environmental factors in REVIC but is less sensitive and has less impact on the estimate. The values for the Tier 1 and Tier 3 inputs were developed by expert judgment (29). The impact of schedule and integration penalties are not addressed in this section. See Findings 5 and 6 for additional information regarding these issues.

Finally, it should be mentioned that the values for all factors can be changed by modifying SASET's calibration file. However, since these values were developed utilizing regression and expert judgment techniques, users should not adjust these values unless they are confident a more reliable estimate will result.

PRICE-S. Other than lines of code, the key cost drivers for PRICE-S estimates include application (APPL), new design (NEWD), new code (NEWC), productivity factor (PROFAC), platform (PLTFM), and complexity (CPLX1). Figures 4-9 through 4-14 are provided to illustrate the impact of these factors on development cost.

At first glance, it does not appear PRICE-S offers as many input options as the other models. This is not the case since many of the key cost drivers are composite descriptors. For example, CPLX1 is a composite of personnel experience, automated tool use, product familiarity, and complicating factors. Likewise, the user selects the PROFAC based on the project's source language, application, and platform.

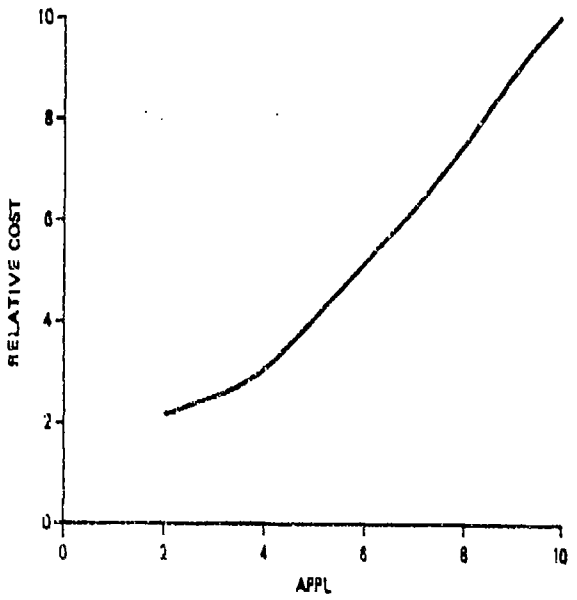


Figure 4-9. Effect of Application on Cost for PRICE-S (27:D-1-55)

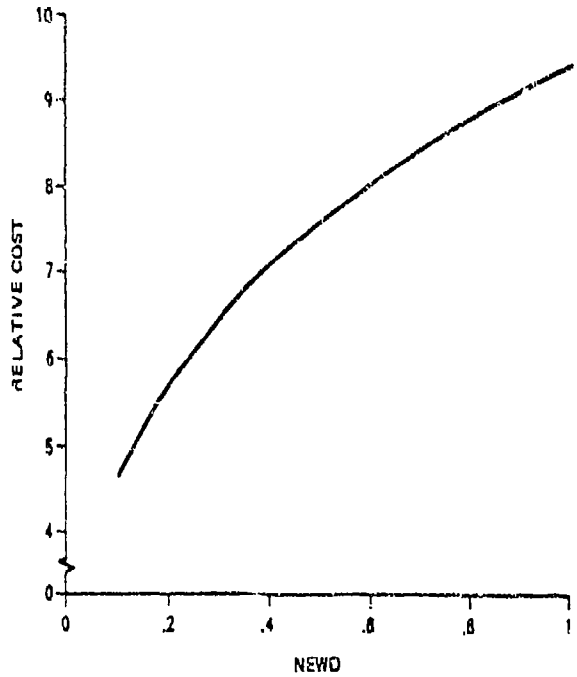


Figure 4-10. Effect of New Design on Cost for PRICE-S (27:D-1-57)

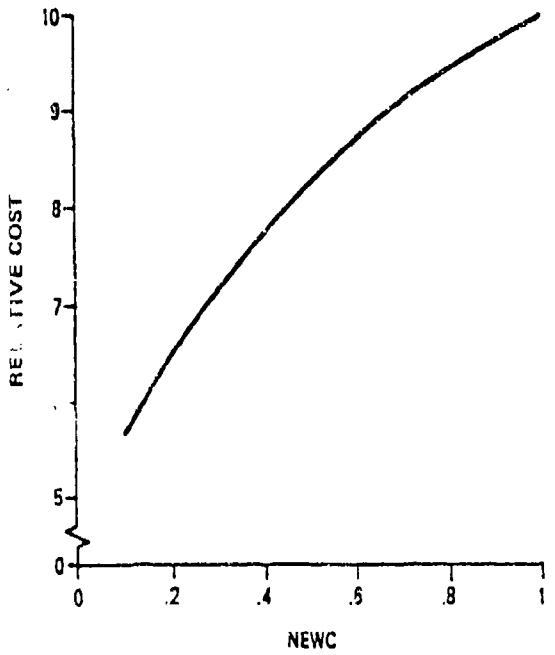


Figure 4-11. Effect of New Code on Cost for PRICE-S (27:D-1-58)

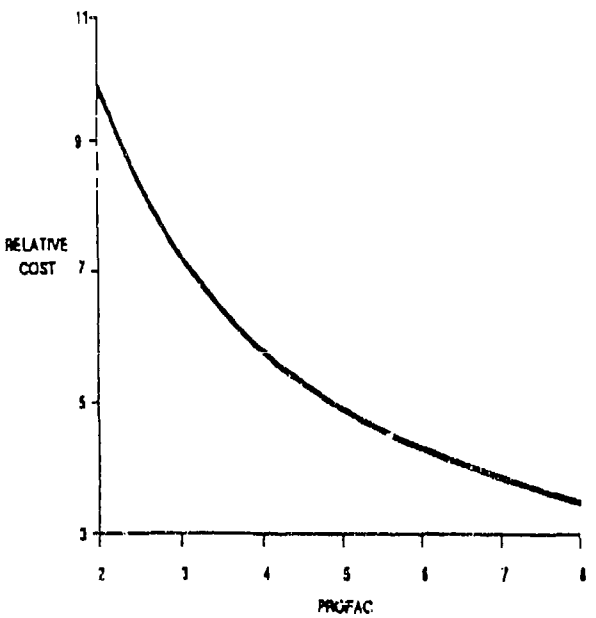


Figure 4-12. Effect of Productivity Factor on Cost for PRICE-S (27:D-1-54)

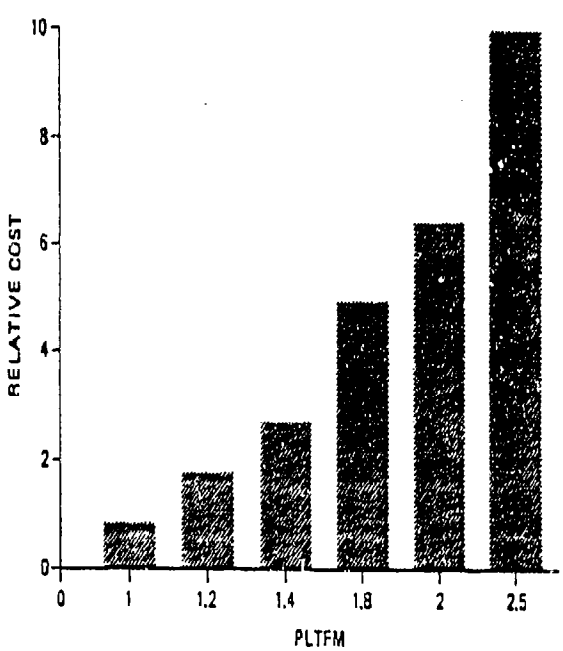


Figure 4-13. Effect of Platform on Cost for PRICE-S (27:D-1-56)

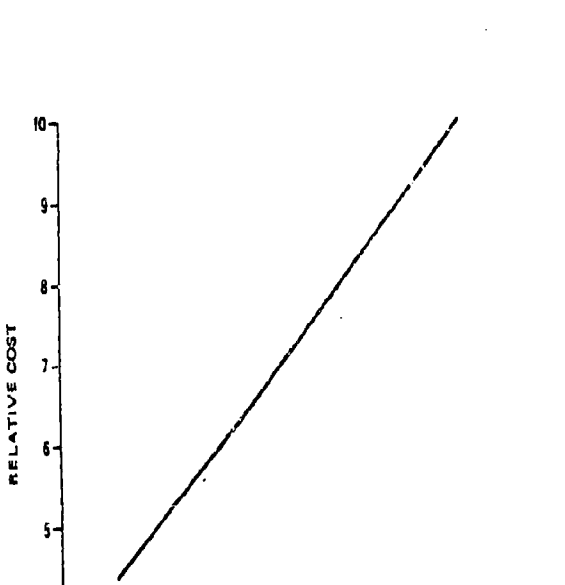


Figure 4-14. Effect of Complexity on Cost for PRICE-S (27:D-1-59)

SEER-SEM. SEER-SEM uses environmental factors to incorporate the influence of the development environment on effort and schedule (19:D-1). Although the model provides default input values based on the knowledge base chosen, SEER-SEM allows the user to adjust approximately 50 factors to more accurately reflect the development project (assuming additional information is available).

According to the SEER-SEM user's manual, the effect of these factors on development effort is a function of the range of values the factor can take (19:D-1). Figures 4-15 through 4-19 are provided to illustrate the impact of many key environmental factors.

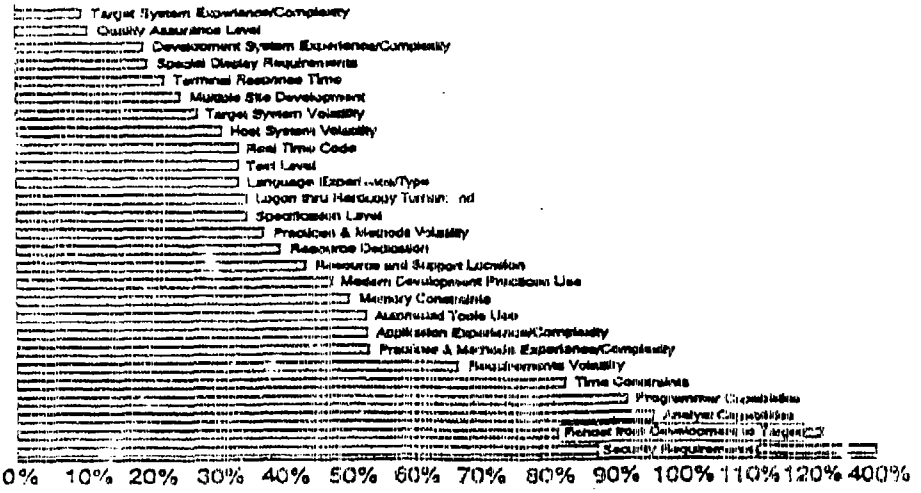


Figure 4-15. Technology and Environment Impacts for SEER-SEM (37:Appendix A)

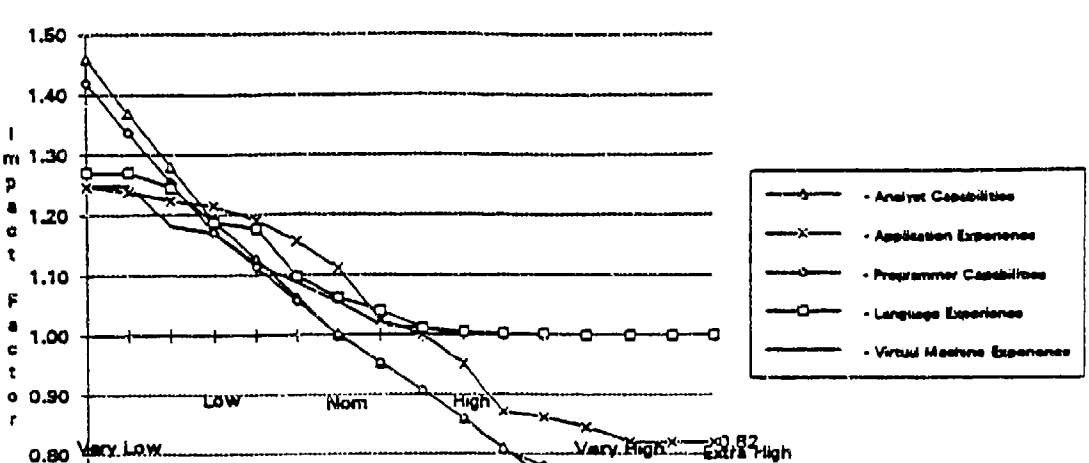


Figure 4-16. Impact of SEER-SEM Inputs for Personnel Capabilities and Experience (37:Appendix A)

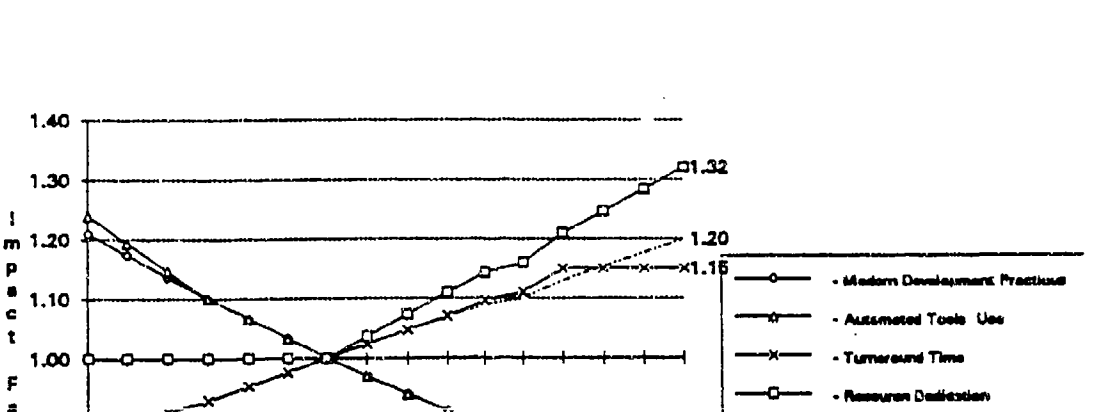


Figure 4-17. Impact of SEER-SEM Inputs for Development Environment (37:Appendix A)

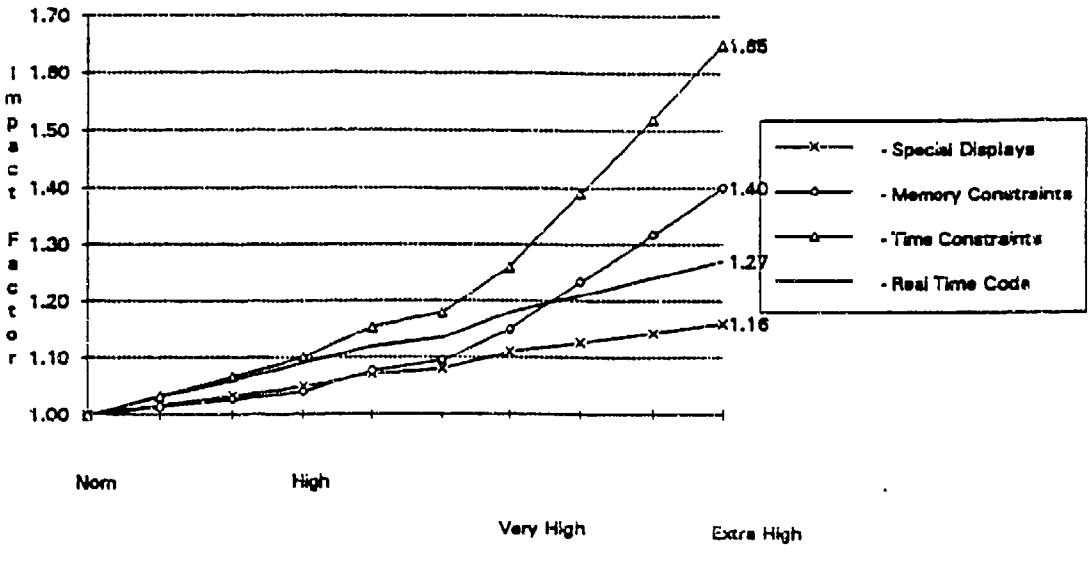


Figure 4-18. Impact of SEER-SEM Inputs for Target Environment (37:Appendix A)

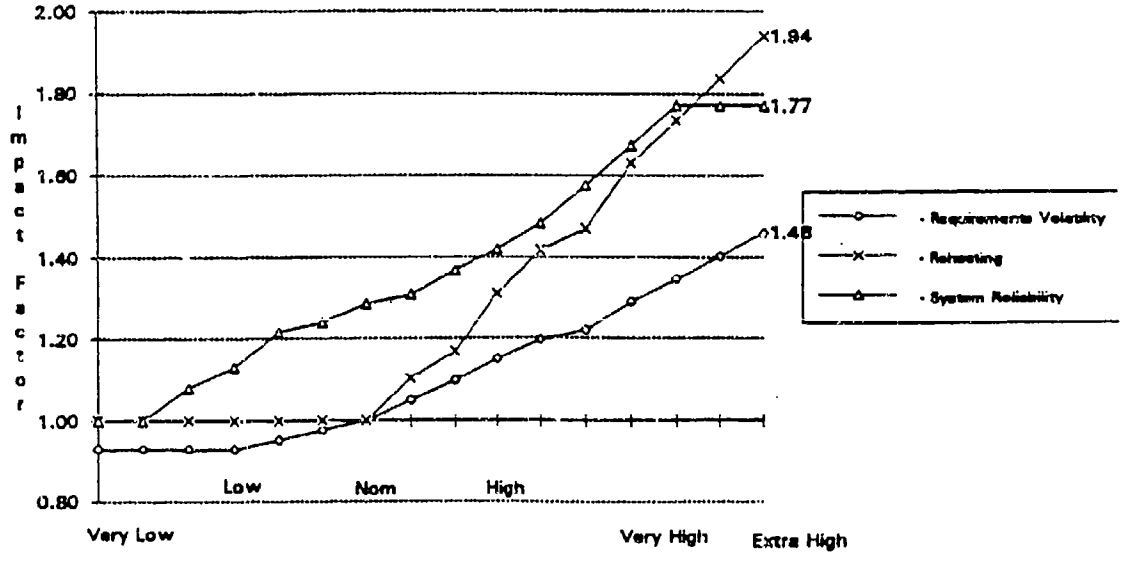


Figure 4-19. Impact of SEER-SEM Inputs for Project Requirements (37:Appendix A)

Finding #5. Implications of Project Size on Model Output

Software development effort is a function of project size as well as numerous environmental factors such as project complexity, target computer system, and development staff capabilities. Since project size tends to have a significant impact on a project's cost, model users should understand the functional relationships between project size and development effort. For example, does the model assume effort increases at an increasing or decreasing rate as CSCI size increases?

To preclude inappropriate use of a model, users should also be aware of its basic limitations. Does the model estimate at the project, CSCI, or CSC level? Is the model limited by the number of CSCIs? Additionally, what is the recommended estimating range for the model? (The recommended estimating range for each model was obtained from model developers/vendors). For instance, if the recommended estimating range for a model is from 5,000 to 100,000 SLOC at the CSCI level, the user should be cautious of estimates based on 130,000 SLOC CSCIs.

Each model was examined to identify the functional relationship between project size and development effort and the impact of breaking large CSCIs into multiple smaller CSCIs. Additionally, each model's coverage regarding CSCIs, CSCs, and CSUs was researched and recommended estimating ranges were verified. Table 4-22 summarizes the results of this review.

Table 4-22. Correlation Matrix of Project Size Relationships

	REVIC	SASET	PRICE-S	SEER-SEM
Functional Relationship Between Size and Effort	Depends on Development Mode	Increases at a Linear Rate	Increases at a Decreasing Rate	Increases at a Increasing Rate
Impact of Breaking Project into Multiple CSCIs	No Impact	Effort Increases	Effort Decreases **Note 1**	Effort Decrease **Note 1**
Integration Costs for Multiple CSCI Projects	No	Yes	Yes	Yes
Estimating Levels Addressed by Model	**Note 2**	CSCIs & CSCs	CSCIs & CSCs	CSCIs, CSCs, & CSUs
Recommended Estimating Range	500 to 130,000 SLOC per data file	500 to 120,000 SLOC per CSCI	5000 to 200,000 SLOC **Note 3**	2000 to 200,000 SLOC per CSCI

Note 1: Effort for individual CSCIs decreases; however, total effort may increase or decrease due to integration costs.
Note 2: Depends on user definition. REVIC estimates modules and does not differentiate between CSCIs, CSCs, or CSUs.
Note 3: Assumes no model calibration. Range will change if model is calibrated to past efforts by the user's organization.

REVIC. REVIC assumes effort increases at a slightly increasing rate as size increases for the organic, semi-detached, and embedded development modes. However, effort increases at a slightly decreasing rate with respect to size for the Ada development mode. These effects are explained by the exponents associated with the equations used to estimate effort. The coefficient and exponent used in each effort equation is dependent on the development mode. REVIC's effort equations are identified in Table 4-23.

Table 4-23. Effort Equations Used by REVIC (20:Sec III, 5)

Development Mode	Effort Equation
Ada	$Effort = 6.8000 * (KDSI)^{0.94}$
Organic	$Effort = 3.4644 * (KDSI)^{1.05}$
Semi-detached	$Effort = 3.9700 * (KDSI)^{1.12}$
Embedded	$Effort = 3.3120 * (KDSI)^{1.20}$

The exponents associated with the organic, semi-detached, and embedded development modes are greater than 1.0 which implies an "increasing at an increasing rate" relationship; whereas, the exponent associated with the Ada development mode is less than 1.0 which implies an "increasing at a decreasing rate" relationship. Figure 4-20 graphically depicts the relationship between size and effort for each development mode based on REVIC's default settings.

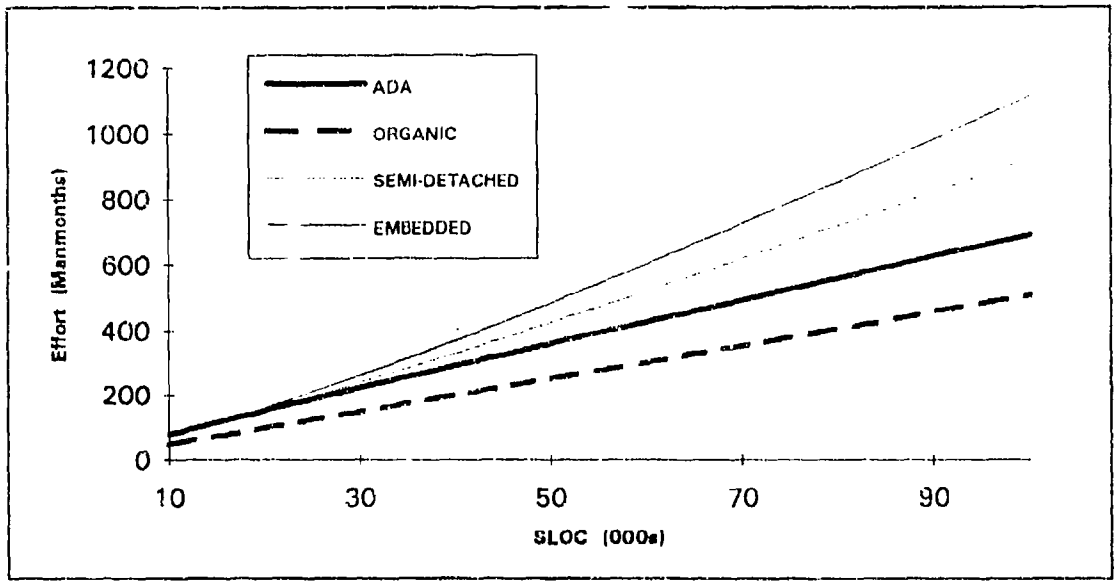


Figure 4-20. Functional Relationship Between Size and Effort for REVIC

Impact of Breaking a Single Large CSCI into Smaller Multiple CSCIs.

REVIC does not differentiate between CSCIs, CSCs, and CSUs. The user inputs size into modules and determines whether a module is a CSCI, CSC, or CSU. Since REVIC does not explicitly add effort for integration between modules, there is no impact on effort and schedule for breaking a project into smaller units in the same session.

REVIC generates the same effort and schedule for a single 100,000 SLOC module or four 25,000 SLOC modules. However, this assertion is true only if all four 25,000 SLOC modules are estimated in the same REVIC file. For example, let's assume four 25,000 SLOC modules based on the embedded development mode will be developed and all default variables are left constant. If the four modules are entered in a single REVIC file, an effort estimate of 1,114.8 manmonths is generated. An estimate based on a single 100,000 SLOC module will yield an equivalent estimate. However, if the user generates a single 25,000 module and multiplies it by four to account for 100,000 SLOC, only 844.8 manmonths are estimated. This difference is caused by the non-linear relationship between size and effort. As size increases, effort increases at an increasing rate and more effort is required.

A project with several modules developed dependently (i.e. run in the same REVIC data file) takes proportionately more time than the same size modules developed independently and run in different REVIC data files (38:2). The exception to this rule if is the Ada development mode is used since its effort equation has an exponent less than one. Consequently, it is recommended that Ada developed modules, which will be integrated with other modules, should be estimated in separate REVIC data files to avoid underestimating effort (34).

Basic Limitations of Model Regarding CSCIs, CSCs, and CSUs. REVIC does not estimate in terms of CSCIs, CSCs, or CSUs. Accordingly, the user must specify modules which adequately describe the effort under consideration.

Recommended Estimating Range for M odel. According to the REVIC model developer, the recommended estimating range is from approximately 500 to 130,000 SLOC per REVIC data file (34). As a result, a project with four 50,000 SLOC CSCIs (totaling 200,000 SLOC) should not be run in the same

data file since the data file size exceeds 130,000 SLOC. In this situation, the user should run the CSCIs in different data files and adjust the default DT&E rate to account for integration costs.

For example, if the user wants to estimate the development and integration costs of three CSCIs run in separate data files, the DT&E default rate should be increased by three percent for each CSCI integration. For this example, the default DT&E rate would be increased from 22% to 28% for each data file. However, this adjustment is not necessary when the Ada Development Mode is selected because the associated equation was calibrated on data which included CSCI integration costs (34).

SASET. SASET assumes effort increases at a linear rate with respect with size. The core of SASET's estimate was previously defined as size, software type, and software class. SASET assigns a value for software type and class which is multiplied by size to estimate development effort. Figures 4-21 and 4-22 illustrate the relationship between size and effort for each software type and software class.

To illustrate how SASET calculates total effort, Figure 4-21 shows that a 100,000 SLOC systems project results in 330,000 manhours. Figure 4-22 shows that a 100,000 SLOC manned flight project results in 300,000 manhours. As a result, SASET will provide an estimate of 660,000 manhours for a 100,000 SLOC manned flight, systems project (assuming all other factors and penalties equal one).

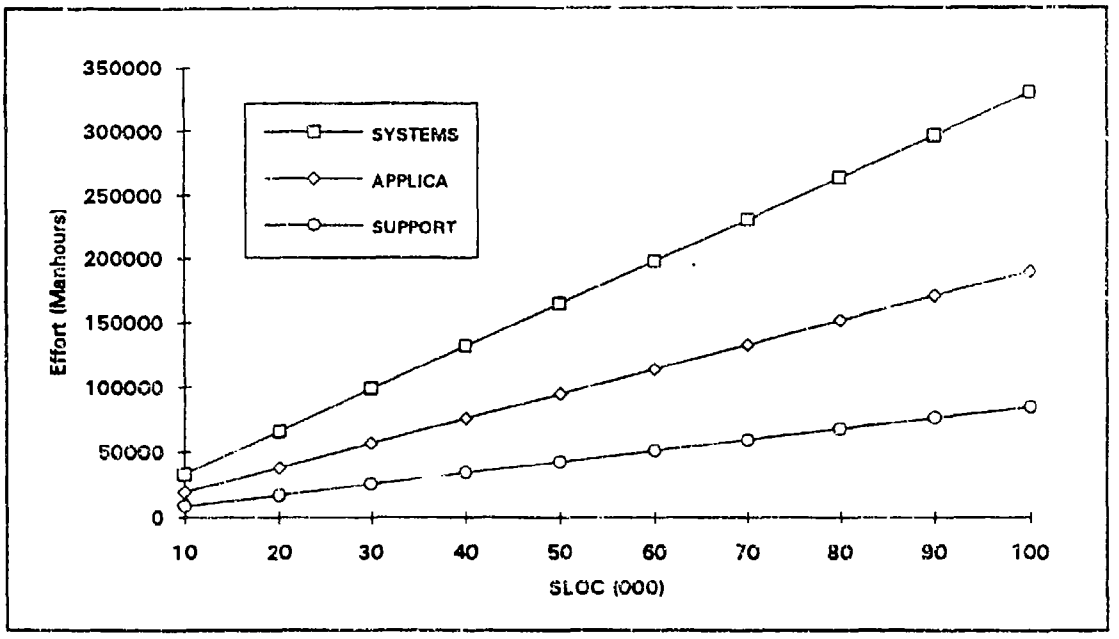


Figure 4-21. Functional Relationship Between Size and Effort for Various Software Types in SASET

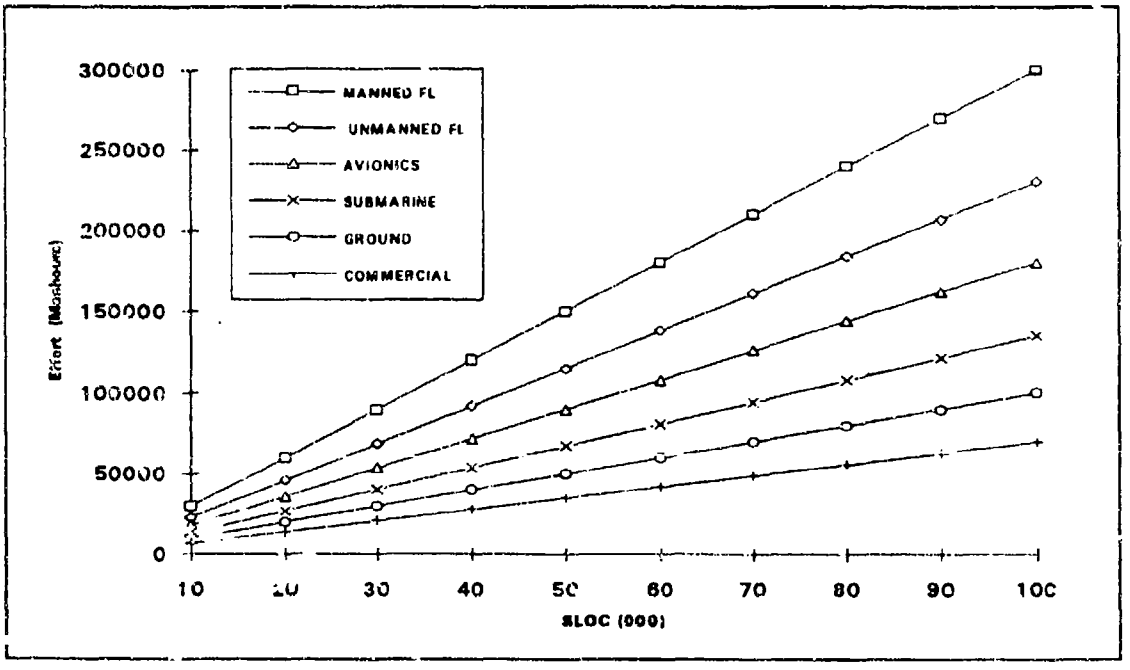


Figure 4-22. Functional Relationship Between Size and Effort for Various Software Classes in SASET

It can be deduced from Figures 4-21 and 4-22 that effort increases at a linear rate with respect to size when all complexity factors (SBM and SSBM) and penalties (schedule and integration) are held constant. However, the rate or slope at which effort increase can greatly vary depending upon the software type and class. As size increases, the differences in effort becomes more pronounced. The reader should note this assumption of linearity assumes that all complexity factors and penalties are held constant. This will rarely be the case when running the model because as the size of the project and CSCIs increase, the project will become more complex.

Impact of Breaking a Single Large CSCI into Multiple Smaller CSCIs. SASET allows the user to break projects into multiple CSCIs; however, total development effort will increase due to the assessment of integration penalties. Integration penalties are only assessed at the CSCI level and are independent of the number of CSCs within a CSCI. SASET assigns an integration penalty of 3 to 12% on the effort for each CSCI, depending on the value of the integration complexity factors. Table 4-24 illustrates the impact of breaking a large CSCI into multiple smaller CSCIs.

Table 4-24. Impact of Breaking Large CSCI into Multiple Smaller CSCIs in SASET

Number of CSCIs	Effort (Simple Integration)	Effort (Very Complex Integration)	Schedule (Simple Integration)	Schedule (Very Complex Integration)
1 100,000 SLOC	170,020 hours	170,020 hours	31.8 months	31.8 months
2 50,000 SLOC	176,666 hours	192,103 hours	35.2 months	35.2 months
4 25,000 SLOC	178,984 hours	194,623 hours	32.0 months	32.0 months
10 10,000 SLOC	182,848 hours	198,825 hours	32.2 months	32.2 months

Assumptions: Size: 100,000 SLOC S/W Type: Applications S/W Class: Ground
Default values for all other inputs.

Although total development effort increases, the majority of the increase occurs when the effort is split from one to two CSCIs. The only reason the effort for four CSCIs is greater than for two CSCIs is because SASET increases the value of the Software Budget Multiplier (calculated from Tier 1 inputs) as the number of CSCIs increases. However, the impact of the increased SBM value is not significant.

The impact of breaking a project down into more CSCIs on schedule is less straightforward. Table 4-24 shows that breaking the project into two CSCIs of 50,000 SLOC generates a schedule of 35.2 months. However, a project of four CSCIs of 25,000 SLOC each generates a schedule of 32 months. This occurs because large CSCIs are the schedule drivers for multiple CSCI projects. In a multiple CSCI project, SASET calculates the schedule for each individual CSCI. The model then extends the schedule for system requirements analysis and integration and testing because it is a multiple CSCI project. For example, the one CSCI of 100,000 SLOC had a schedule of 31.8 months. If that CSCI was part of a multiple CSCI project, SASET would still generate a schedule of 31.8 months for that CSCI but would extend the schedule for system requirements, system integration, and testing. This explains why the project with the two larger 50,000 SLOC CSCIs takes more time than the project with four smaller CSCIs.

With this principle in mind, one may question why the ten CSCI project has a longer schedule than the four CSCI project. This occurs because SASET assumes requirements analysis on each individual CSCI is started one month after the previous CSCI (i.e., for a 10 CSCI project the requirements analysis on the 10th CSCI would be started 9 months after the first CSCI). As a result, even though the schedule for each individual CSCI in the 10 CSCI project is shorter

than the four CSCI project, the schedule for the project is slightly longer because work on the tenth CSCI is started 9 months after the first CSCI. In summary, breaking a project into multiple CSCIs will increase schedule. However, the amount of increase is a function of the size of the largest CSCI in the project and, to a lesser extent, the number of CSCIs in a project (29).

Basic Limitations of Model Regarding CSCIs, CSCs, and CSUs. SASET allows the user to describe the project in terms of CSCIs and CSCs. The number of CSCIs is limited by the amount of available conventional computer memory. There is no limit to the number of CSCs for a CSCI (29).

Recommended Estimating Range for Model. SASET allows the user to input CSCIs as small as 1 SLOC and as large as 1.2 million SLOC. However, a Martin Marietta software engineer stated he would be cautious generating estimates for CSCIs smaller than 500 SLOC or larger than 120,000 SLOC (29).

PRICE-S. PRICE-S generally assumes development effort increases at a decreasing rate as project size increases. However, according to a PRICE-S model developer, the model estimates effort in a relatively linear manner between 40,000 to 60,000 lines of code (30). PRICE-S uses composite descriptors to model project size. The composite descriptors are primarily a function of Application, New Design, and New Code. Figures 4-9 through 4-11 graphically depict the relationship between these factors and relative cost for the project.

Impact of Breaking a Single Large CSCI into Multiple Smaller CSCIs. The model allows the user to break large projects into multiple CSCIs. Development time for individual CSCIs decreases as CSCI size decreases. However, total development effort for multiple CSCIs may be greater or less than the effort associated with a single, large CSCI depending on how much system integration effort is required. PRICE-S calculates effort for system integration based on

contributions from each CSCI. This contribution is a function of Modified Source Lines of Code, weighted Application, Platform, Utilization, Productivity Factor, External Integration Factor, and schedule (27:D-1-62).

Table 4-25 illustrates the impact of breaking a large CSCI into multiple smaller CSCIs.

Table 4-25. Impact of Breaking Large CSCI Into Multiple Smaller CSCIs in PRICE-S

Number of CSCIs	CSCI Development Effort (man-months)	System Integration Effort (man-months)	Total Project Effort (man-months)
1 100,000 line CSCI	1226.8	0.0	1226.8
[illegible]0,000 line CSCIs	1168.2	106.9	1275.2
4 [illegible]5,000 line CSCIs	1112.3	107.3	1219.7
10 [illegible]0,000 line CSCIs	1066.4	107.1	1174.1

Assumptions				
	Size:	100,000 SLOC	New Design:	1.0
	Platform:	1.2	New Code:	1.0
	Application:	6.16	Productivity Factor:	5.0
	SDR:	Mar 94 (9 months in future)	Language:	Ada
	INTEGE/INTEGI:	.5	Default values for all other inputs	

Basic Limitations of Model Regarding CSCIs, CSCs, and CSUs. PRICE-S allows the user to describe projects in terms of CSCIs and CSCs. The number of CSCIs is limited only by the amount of available memory on the computer used to generate the estimate (30). Within the PRICE-S model, CSCs are associated only with development CSCIs. A maximum of 25 CSCs are permitted for each CSCI.

Recommended Estimating Range for Model. Valid SLOC inputs for PRICE-S ranges from one to 999,999,999 lines of code. However, two factors may limit the size of large projects. Specifically, the model will not allow any

development phase to exceed five years and the total development effort cannot exceed 20 years (30). According to PRICE-S personnel, a general rule is to divide the project into logical, manageable CSCIs (30). The recommended estimating range is from approximately 5000 to 200,000 SLOC; however, the model has been successfully used for significantly larger CSCIs (39). For small projects (less than 1000 lines of code), it was suggested that users bypass the model and simply applying company-specific measures such as lines of code per hour to calculate the cost (30).

SEER-SEM. SEER-SEM assumes development effort increases at an increasing rate as project size increases. This effect is explained by the exponent associated with a key equation used to estimate development effort:

$$Effort = 0.4 S_e^{1.2} D^{0.4} C_{tb}^{-1.2} f^{1.2}$$

where:

S_e = Effective Size measured in SLOC
D = System Complexity
C_{tb} = Basic Technology Constant
f = Composite Adjustment Factor obtained from several environmental factors

The exponent associated with "Effective Size" is greater than 1.0 which implies an "increasing at an increasing rate" relationship. Figure 4-23 graphically depicts the relationship between size and effort for four development modes based on SEER-SEM's default settings for a ground-based radar system.

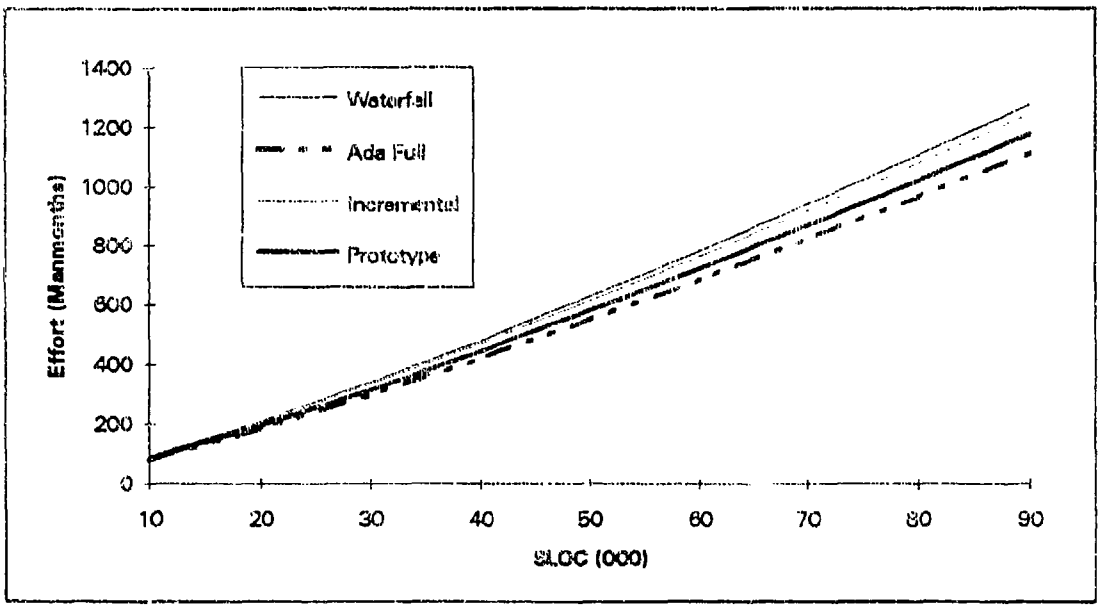


Figure 4-23. Functional Relationship Between Size and Effort for SEER-SEM

Impact of Breaking a Single Large CSCI into Multiple Smaller CSCIs. SEER-SEM allows the user to break large projects into multiple CSCIs. Development time for individual CSCIs decreases as CSCI size decreases if no CSCI integration is required. However, total development effort for multiple CSCI projects may be greater or less than the effort associated with a single, large CSCI project depending on the level of CSCI integration. Table 4-26 illustrates the impact of breaking a large CSCI into multiple smaller CSCIs.

Basic Limitations of Model Regarding CSCIs, CSCs, and CSUs. SEER-SEM allows the user to describe projects in terms of CSCIs, CSCs, and CSUs. The number of CSCIs, CSCs, and CSUs is limited only by the amount of available memory on the computer used to generate the estimate (32)

Table 4-26. Impact of Breaking Large CSCI Into Multiple Smaller CSCIs in SEER-SEM

Number of CSCIs	Effort (No CSCI Integration)	Effort (Increasing Complex Integration)	Schedule (No CSCI Integration)	Schedule (Increasing Complex Integration)
1 100,000 SLOC	1448.1	1448.2	51.4	51.4
2 50,000 SLOC	1260.6	1287.0	39.0	39.5
4 25,000 SLOC	1097.4	1166.4	29.5	30.8
10 10,000 SLOC	913.7	1086.0	20.5	23.1

Assumptions:

Size:	100,000 SLOC	New Code:	100%
Platform:	Ground	Development Method:	Waterfall
Application:	Radar	Development Standard:	2167A

Default values for all other inputs

Recommended Estimating Range for Model. SEER-SEM allows the user to input CSCIs as small as 1 SLOC and as large as 100 million SLOC. However, the model developer stated the recommended estimating range is from 2,000 to 200,000 SLOC at the CSCI level. He further stated SEER-SEM was designed to model and simulate reality and that users should specify the CSCI size that is anticipated for the actual software development (33). According to the model developer, the upper bound was established since "software engineering studies have shown the efficient range for CSCIs is less than 100,000 lines and that CSCIs over 200,000 tend to never get completed properly" (33). A SEER Technologies support technician indicated the most current version of SEER-SEM is not as limited at the low end when using the function sizing mode (32). (Note: This effort did not examine the function sizing capabilities of SEER-SEM).

Finding #6. Impact of Schedule Compression and Extensions

Software cost models usually follow one of two "schools of thought" when calculating cost and schedule estimates: minimum development time or minimum development effort. Minimum development time is the shortest time in which the developer can complete the project while simultaneously satisfying size, staffing, and other project constraints (19:5-6). Estimates based on the minimum development time concept are expected to result in higher total development costs since the project requires maximum effort to satisfy the minimum development time constraint. Conversely, the minimum development effort concept does not constrain the schedule and seeks to minimize total development effort based on an "optimal" schedule generated by the model.

This distinction is important since it impacts how models account for schedule compression and stretchouts. For example, users cannot compress the schedule generated by a minimum development time model because (by definition) the schedule is already at a minimum. Software cost models also have different assumptions regarding schedule inefficiencies. A schedule inefficiency occurs when the user defines a development schedule that differs from the "optimal" schedule generated by the model. Some models assume total development effort increases when the schedule is stretched out; whereas other models assume total development effort decreases. These differences in assumptions can have a dramatic impact on estimates produced by the models. As a result, users should understand how changes in the development schedule will impact the project's cost.

The models were reviewed to determine if schedule compression and extensions were permitted. Model assumptions regarding schedule inefficiencies and their impact were also identified. Table 4-27 summarizes this review.

Table 4-27. Impact of Schedule Compression and Extensions

	REVIC	SASET	PRICE-S	SEER-SEM
Does Model Allow Schedule Compression/Extensions	Yes	Yes	Yes	Yes
Impact of Compressed Schedules on Total Effort	Increases Effort	Increases Effort **See Note 1**	Increases Effort	Decreases Effort **See Note 2**
Impact of Extended Schedules on Total Effort	Increases Effort	Increases Effort	Increases Effort	creases Effort

Note 1: SASET does not allow the user to compress the schedule for multiple CSCI projects.

Note 2: SEER-SEM calculates a minimum development time schedule; however, schedule can be compressed by reducing the probability of the most likely estimate from 50% to lower levels such as 30% or 40%.

REVIC. REVIC allows both schedule compression and extensions. The user may compress REVIC's nominal schedule up to 75% or infinitely extend the schedule. No schedule penalties are applied if the user-defined schedule is within -5% or +30% of REVIC's nominal schedule. However, total development effort will always increase if the user-defined schedule is outside this range. REVIC multiplies the total development effort by a schedule penalty factor based on the amount of schedule compression or extension. Table 4-28 identifies the schedule penalty factors assessed by REVIC.

Table 4-28. Schedule Penalty Factors for REVIC (1:467)

User-Defined Schedule	Penalty Factor
75% < REVIC's Nominal Schedule < 85%	1.23
85% < REVIC's Nominal Schedule < 95%	1.08
95% < REVIC's Nominal Schedule < 130%	1.00
130% < REVIC's Nominal Schedule < 160%	1.04
160% < REVIC's Nominal Schedule	1.10

REVIC's schedule penalties are discrete and assessed in a step function manner. For example, if the schedule is compressed to 85% of REVIC's nominal schedule, an 8% schedule penalty is applied. However, if the schedule is compressed to 84%, the schedule penalty jumps to 23%. No schedule extension penalties are assessed unless the user-defined schedule is greater than 125% of REVIC's nominal schedule. The maximum schedule penalty for extending the schedule is 10%. Due to the step function approach REVIC uses to assess schedule penalties, users should be aware of these special regions when compressing or extending schedules generated by REVIC.

SASET. The user may compress the schedule to 50% of the SASET's nominal schedule for single CSCI projects; however, no schedule compression is permitted for multiple CSCI projects. The schedule can be infinitely extended for single and multiple CSCI projects.

If the user specifies only a start date, SASET calculates the development effort based on "optimal" schedule and no schedule penalties are assessed. If the user inputs both a start and finish date for the project, SASET compares the user-defined schedule to the schedule generated by the model and assesses penalties for any schedule inefficiencies.

For example, assume the user inputs a 48 month development schedule and SASET calculates an optimal schedule of 60 months. The ratio of the user-defined schedule and optimal schedule is 0.80 (48 months / 60 months). SASET refers to an internal lookup table and assesses a schedule penalty based on the ratio. For this example, a ratio of 0.8 corresponds to a penalty factor of 1.10 which increases total development effort by 10%.

Schedule compression has a much more pronounced impact on the estimate than extending the schedule by the same amount. Compressing the

schedule to 50% of the optimal schedule adds 25% to the estimate; however, extending the schedule by the same amount adds only 10% to the schedule. Figure 4-24 illustrates the schedule penalties associated with various schedules.

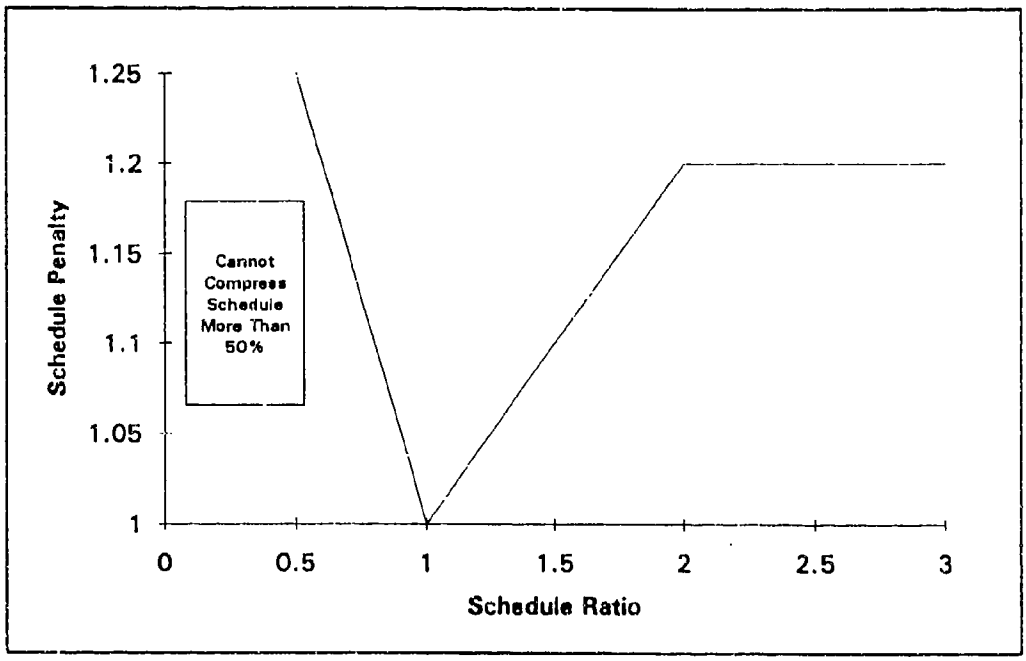


Figure 4-24. Schedule Penalty Factors for SASET

SASET assigns schedule penalties somewhat differently for multiple CSCI projects. Schedule penalties are assessed if the user-defined schedule differs from the optimal schedule. However, schedule penalties are assessed at the project level and not at the CSCI level. Although SASET allows schedule extensions, schedule compression is not permitted for multiple CSCI projects. Users can work around this limitation by defining several single CSCI projects and compressing the schedule of each CSCI. However, this approach will not include integration costs since SASET does not calculate integration penalties for single CSCI projects.

SASET also assigns a schedule penalty if the sequence of CSCIs is less than optimal. The model assumes the optimal sequence of CSCIs is from the largest to smallest CSCI. For example, if there are three CSCIs with sizes of 30,000, 50,000, and 70,000 SLOC; the optimal sequence is 70,000, 50,000, and 30,000 SLOC. A sequence specified in any other order is less than optimal and a schedule penalty is assigned (29). The user may specify a CSCI sequence appropriate for the development project; otherwise, the model assumes the optimal sequence.

PRICE-S. The user may compress or extend the model's reference schedule by entering activity milestone dates different from those calculated by the model. PRICE-S assumes all schedule inefficiencies result in increased project costs; however, no penalties are assessed if the user-defined schedule is within +/- 10% of the model's reference schedule.

PRICE-S initially calculates an internal reference schedule based on the CPLX1 (Complexity 1) input and the SDR or SSR date. This reference schedule is the "normal" development schedule when no time constraints are present and serves as a reference for estimating the added costs when schedule constraints are imposed (31:5).

PRICE-S differs from REVIC and SASET in that the user may compress or extend specific development phases as well as total development time. This is accomplished by entering user-defined dates for key development milestones such as the PDR, CDR, or PCA. PRICE-S then adjusts the reference schedule and costs to account for the effects of user-defined dates. Schedule penalties associated with user-defined dates can be removed by setting the Schedule Phase Multiplier (SMULT) equal to zero for that phase. Like SASET, schedule compression has a much more pronounced impact on the estimate than

extending the schedule by the same amount. Figure 4-25 illustrates the effects of schedule constraints on cost for PRICE-S.

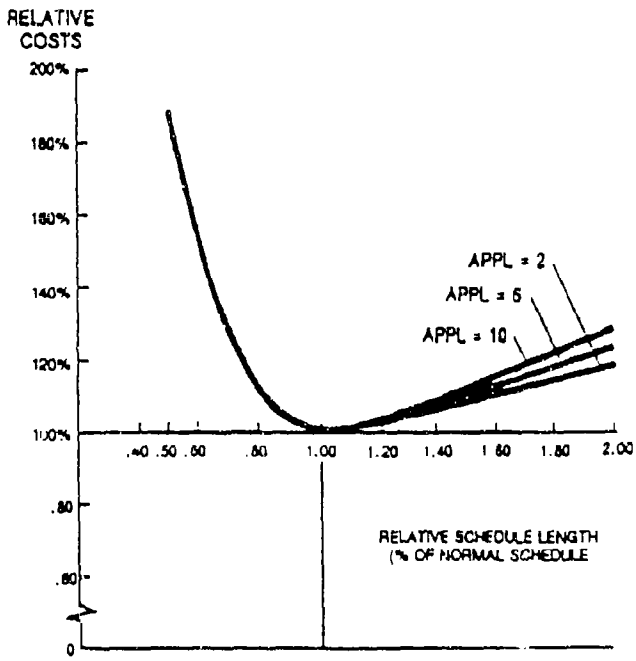


Figure 4-25. Effect of Schedule Constraints on Cost for PRICE-S (27:D-1-60)

SEER-SEM. Although SEER-SEM calculates the optimal (minimum) development schedule, users can "compress" the schedule by reducing the most likely estimate from 50% to a lower value (33). This is accomplished by adjusting the "Probability' parameter. Schedule compression results in less total development time versus the standard minimum development time solution; however, the likelihood of completing the project on time also decreases as the "Probability" parameter is reduced.

The model allows the user to infinitely extend the schedule. SEER-SEM differs from the other models regarding schedule extensions. While REVIC,

SASET, and PRICE-S assume total development time increases if the user-defined schedule exceeds the model's reference schedule, SEER-SEM assumes total development effort decreases when the user-defined schedule is longer than minimum development time solution. This effect is due in part to the model's assumption regarding optimum staffing rates. When the schedule is stretched out, the model assumes staffing levels can be lowered resulting in lower total development costs.

Estimates generated by SEER-SEM are very sensitive to small changes in the development schedule. Table 4-29 illustrates the impact of stretching out full-scale implementation of a 100,000 line ground-based radar project.

Table 4-29. Impact of Stretching Out Full-Scale Implementation for SEER-SEM

	Baselines	+3 months	+6 months	+12 months	+24 months
Development Effort (manmonths)	1448	1227	1058	810	518
Project Schedule (months)	51.4	55.9	60.2	68.8	85.9

For the SEER-SEM model, total development effect decreases significantly as the development schedule is extended. Since the model is based on the Rayleigh curve, the user can theoretically extend the schedule infinitely and total development effort will approach zero. However, there are several practical limitations when performing schedule tradeoffs. Most importantly, it is unrealistic to assume a project can be stretched out indefinitely since such projects would never be undertaken. According to the SEER-SEM user manual, the maximum development schedule should not exceed 60 months from Software Requirements Analysis through CSCI Integration and Testing.

Phases prior to and subsequent to these phases are in addition to the 30 month schedule (19:5-2).

Users should also be aware of potential "warning signs" which indicate the schedule has been stretched out too far. According to SEER Technologies personnel, three factors should be monitored. If the "Peak Staff" drops below five persons or the "Effective Complexity" rating drops below 4.0, the user has probably extended the schedule beyond the feasible region (32, 36). Additionally, the user has probably defined an unrealistic set of parameters if the "Basic Technology" or "Effective Technology" are not within a range of 1,000 to 10,000 (36). Future versions of SEER-SEM will address the impact of unrealistic schedule extensions by providing constraints for minimum staffing levels (32).

The model developer also provided the following insights when stretching the development schedule.

> Stretch the schedule only if you will manage the project that way from the beginning (they do this in Europe ... much less in the DoD where minimum time is the norm). Never stretch the schedule more than 25 - 40% (unless you have a real case where the project will be managed that way). Never stretch the schedule to the point where the average staff is less than about 2 - 3 people (peak about 5 people) unless you know the people. (33)

Finding #7. Distinctive Characteristics of Model Data Bases

REVIC. With the exception of the Ada Development Mode, REVIC was calibrated with "calibration points" provided by Hughes Electro-optical Data Systems Groups (EDSG) (34). The size and characteristics of the EDSG data base were unknown; however, all of the projects were government programs.

According the model developer, the Ada Development Mode equation was calibrated by Bob Jeff of the former Air Force Contract Management Division (34). The data base consisted of approximately 20 Ada CSCIs extracted from Boehm's Ada COCOMO data base and a large Army project. The Army project was a 1.2 million line effort that was broken down into multiple, smaller CSCIs (34). REVIC's equations were validated on a data base of 281 completed contracts with software involvement from Rome Air Development Center (34).

SASET. SASET's equations are based on a data base of 500 CSCIs made up of Martin Marietta's projects and others it gathered from industry (29). More than 90% of the data base consisted of embedded military applications. Most of the CSCIs were programmed in third generation languages (predominantly FORTRAN) and some Assembly code. The size of the CSCIs ranged from 700 to 90,000 lines of code with an average of 35,000 to 45,000 lines of code (29).

PRICE-S. The PRICE-S model was developed on project data obtained from a variety of commercial developers such as Lockheed, Raytheon, and General Dynamics (30). Currently, all projects used in the model's equations are commercial in nature. However, according to PRICE-S personnel, three software data bases are maintained (30). The initial data base is the one on which the model equations were developed. The second data base is used to

evaluate the model's equations and the third is used by operational personnel to test new and innovative changes in the software development area. For example, information related to object-oriented programming is collected and retained in the third data base for potential use in later versions of PRICE-S (30).

The model equations are evaluated yearly; however, this does not necessarily mean the equations are changed that often. Revisions and enhancements are made as necessary. All commonly recognized programming languages are represented by projects in the data base.

SEER-SEM. The internal equations of SEER-SEM are based on a mathematical model developed by Dr. Randall W. Jensen plus numerous extensions and enhancements to reflect changes in software development practices and techniques (33). SEER-SEM also relies on "knowledge bases" to provide detailed inputs for model estimates. According to the model's user manual, a knowledge base provides detailed input values based on a general description of the program (37:Appendix A). The user enters four inputs describing the program (Platform, Application, Development Method, and Development Standard) and the model selects the appropriate setting for the detailed inputs. The user may modify any of the detailed parameters if additional information is available.

The knowledge bases are heuristic in nature and frequently updated to reflect the latest software development trends (36). Organizations are evaluated to determine what changes are occurring in the software development environment and then knowledge bases are adjusted to account for these changes. For example, if today's software developers use more modern development practices and automated tools than in the past, the inputs for the

knowledge bases are adjusted to reflect these changes (36). Both commercial and military projects were used to develop SEER-SEM's knowledge bases and all commonly used programming languages (FORTRAN, Assembly, Ada, etc.) are represented (32).

Finding #8. Results of Baseline Test Case

A simple baseline test case was developed to gain a better understanding of why the models generate different estimates. The purpose of the test case was not to quantify the differences between model estimates, but to explain the underlying reasons for the differences. Additionally, it was hoped this effort would result in greater insight into the feasibility of normalizing the models and the difficulties associated with normalization efforts.

Test Case Limitations. Due to the time constraints associated with this research effort, a simple, generic development project was proposed. This approach is a significant limitation since the case does not represent the actual complexity associated with a real development project. However, in the researchers' opinion, it was deemed adequate to illustrate the differences between model estimates.

Test Case Scenario. The following information provides details regarding a hypothetical software development project. This hypothetical case is a modified version of a baseline test case presented in Sydney Rowland's "A Comparative Analysis of the GE PRICE-S and the CEI System-4 Cost Estimating Models" (40). The assumptions for the case include:

1. The development project will result in an aircraft avionics system. Tailored DoD-STD-2167A documentation is required.

2. The waterfall life cycle approach is used and integration complexity for all CSCIs is average.

3. The project consists of three CSCIs which are referred to as CSCI1, CSCI2, and CSCI3. Each CSCI is 100% New Design and New Code.

4. CSCI1 is programmed in Ada and consists of two CSCs developed by average personnel with nominal software tools. The first CSC has 20,000 SLOC and the second CSC consists of 30,000 SLOC.

5. CSCI2 is programmed in Assembly by above average programmers with nominal software tools. The CSCI has 80,000 SLOC.

6. CSCI3 is programmed in Ada by average personnel with nominal software tools. The CSCI has 45,000 SLOC.

7. The project is estimated from System Software Requirements Analysis through System Test.

8. The development teams work 152 manhours per month.

Model Inputs. Where possible, suggestions from model vendors and developers were obtained during development of the baseline test case. Unless noted otherwise, the researchers used default input values for each model. See Table 4-30 for a summary of model inputs used for the baseline test case and Appendix B for detailed model input sheets.

REVIC. Each CSCI was loaded into a separate data file to avoid exceeding the recommended estimating range of the model. As a result, three data files were created for the REVIC estimate. The first data file, which represented CSCI1, consisted of two CSCs. Integration was not added to CSCIs 1 and 3 because the Ada Development Mode equation was calibrated on CSCIs which includes integration effort. The embedded development mode was used for CSCI2 since the entire CSCI was programmed in Assembly. The DT&E parameter for CSCI2 was also changed from 22% to 28% to account for integration with the other CSCIs.

Table 4-30. Summary of Key Model Inputs for Baseline Test Case

	REVIC	SASET	SEER-SEM	PRICE-S
CSCI1	Separate Data File for Each CSCI 2 CSCs loaded into into this file CSC1: 20K SLOC CSC2: 30K SLOC Ada Dev Mode RELY: HI Default values for all other inputs	One Data File for Total Estimate 2 CSCs CSC1: 20K SLOC CSC2: 30K SLOC S/W Class: Avionics S/W Type: Application S/W Documentation: Complex Man Interaction: Complex Default values for all other inputs	One Data File for Total Estimate 2 CSCs CSC1: 20K SLOC CSC2: 30K SLOC Avionics Platform Flight Application Ada Development Method 2167A min Development Standard Default values for all other inputs	One Data File for Total Estimate 2 CSCs CSC1: 20K SLOC CSC2: 30K SLOC PROFAC: 5.00 APPL: 5.50 PLTFM: 1.80 SSR Date: 894 INTEGE & INTEGI: 0.50 CPLX1: 1.00 Default values for all other inputs
CSCI2	Separate Data File Embedded Dev Mode PCAP: HI LEXP: HI RELY: HI DT&E: 28% 80K SLOC Default values for all other inputs	S/W Class: Avionics S/W Type: Application S/W Experience Simple Programming Language: Complex S/W Documentation: Complex Man Interaction: Complex 80K SLOC Default values for all other inputs	Avionics Platform Flight Application Waterfall Development Method 2167A min Development Standard Programmer Capability: Nom, HI, VHI Programmer Lang Exp: HI, VHI, EHI Language Type: HI, HI, VHI 80K SLOC Default values for all other inputs	PROFAC: 5.00 APPL: 5.50 PLTFM: 1.80 SSR Date: 894 INTEGE & INTEGI: 0.50 CPLX1: 0.80 80K SLOC Non-Executable SLOC: 0 Default values for all other inputs
CSCI3	Separate Data File Ada Development Mode RELY: HI 45K SLOC Default values for all other inputs	S/W Class: Avionics S/W Type: Application S/W Documentation: Complex Man Interaction: Complex 45K SLOC Default values for all other inputs	Avionics Platform Flight Application Ada Development Method 2167A min Development Standard 45K SLOC Default values for all other inputs	PROFAC: 5.00 APPL: 5.50 PLTFM: 1.80 SSR Date: 894 INTEGE & INTEGI: 0.50 CPLX1: 1.00

Default values were used for all inputs with the exception of PCAP, LEXP, RELY, and DT&E. PCAP and LEXP were adjusted to HI for CSCI 2 to account for improved programmer capabilities. The model developer recommended adjusting RELY to HI for the baseline case (34). (See Appendix B for model input sheets.)

SASET. All CSCIs were run in the same file. The avionics S/W Class was selected and the S/W Type was Applications. Default input values were used for all inputs with the exception of software documentation, man interaction, software experience, and programming language. Software documentation was set to COMPLEX since tailored DoD-STD-2167a standards apply. The man interaction input was adjusted to COMPLEX to account for the avionics S/W Class. For CSCI2, the software experience input was set to SIMPLE to account for improved programmer capabilities with regard to Assembly programming; whereas, the programming language was adjusted to COMPLEX to reflect the use of the Assembly language. (See Appendix B for model input sheets.)

PRICE-S. Default values were used for all inputs with the exception of PROFAC, APPL, SSR date, and CPLX1. Since the model was not calibrated to a specific development environment, subjective decisions were made regarding several key input parameters such as PROFAC and APPL. After reviewing the PRICE-S Reference Manual and related documentation, values for the PROFAC and APPL were 5.00 and 5.50, respectively. INTEGE and INTEGI inputs were set to 0.50. Lacking any information to the contrary, these inputs were deemed adequate for this effort; however, users should be aware of the sensitivity of these parameters and their impact on the resulting estimate.

The SSR date was adjusted to ensure no schedule penalties were

encountered. CPLX1 was adjusted for CSCI2 to account for differences in programmer capabilities. (See Appendix B for model input sheets.)

SEER-SEM. The avionics platform and flight application knowledge bases were selected to represent the avionics flight system. The Ada Development method was selected for CSCIs 1 and 3; whereas, the waterfall development method was selected for CSCI 2. The development standard was 2167Ammn for all CSCIs. Knowledge base inputs for CSCI2 were adjusted to account for differences in programmer capabilities. (See Appendix B for model input sheets.)

Summary of Model Estimates. Model estimates for each CSCI and the total project are summarized in Table 4-39.

Table 4-31. Baseline Test Case Results for Each Model in Manmonths

Estimate Level	REVIC	SASET	PRICE-S	SEER-SEM
CSCI 1	416.0	1104.2	1429.5	719.4
CSCI 2	837.2	545.3	520.2	1013.0
CSCI 3	376.7	993.8	1211.8	634.0
Project	1648.5	2643.3	3161.5	2366.4

Discussion of Results. SASET and PRICE-S provided significantly higher estimates for CSCIs 1 and 3 compared to REVIC and SEER-SEM. However, SASET and PRICE-S provided lower estimates for CSCI2. The following paragraphs discuss some of the underlying reasons for these differences.

Excluded Development Phases/Cost Elements. REVIC does not include the System Requirements/Design phase or the necessary effort to meet DoD-STD-2167A documentation requirements. These factors contribute to REVIC's

low estimates for CSCIs 1 and 3. When using REVIC as a cross-check to a different model, the user would have to estimate and add additional effort for these omissions.

Treatment of Different Languages. It seemed unusual that SASET and PRICE-S provided much lower estimates for CSCI 2 despite being higher for CSCIs 1 and 3. This is due in part to their treatment of the Assembly language. SASET and PRICE-S are more sensitive to language differences than REVIC or SEER-SEM. SASET and PRICE-S assume programmers can write more lines of Assembly code than Ada code for a given period of time. For example, SASET converted the 80,000 lines of Assembly to 26,667 lines of a "Equivalent New HOL" and based its estimate on this number.

REVIC and SEER-SEM are less language dependent and do not make similar adjustments for the Assembly language. For example, with the exception of using the Ada Development Mode for Ada code, REVIC does not differentiate between languages. As a result, it may be appropriate to research a proper software language conversion metric to account for language differences when using REVIC.

Sensitivity of Inputs/Cost Drivers. Even with this simple scenario, there was obvious subjectivity regarding appropriate input values. For example, subjective values for PROFAC and APPL were used for PRICE-S. Yet, the model is very sensitive to small changes in these parameters and small favorable adjustments can significantly lower the estimate. With REVIC, the default value for MRES was used; however, based on the model's definition for this input, a case could be made to adjust this value (which would have dramatically increased REVIC's estimates). Similar points could be made about various inputs for SASET and SEER-SEM. Thus, a different person could use

the same test case and generate completely different estimates.

In the test case, it appears that one of the reasons why SASET and PRICE-S generally provide higher estimates is their sensitivity to the Platform selected. During our research efforts and sensitivity analyses with the models, we noted adjustments to the PRICE-S Platform input and SASET Class of Software input significantly impacted the estimates. For example, SASET's equations reveal that an avionics project estimate will be approximately 80% greater than a ground based system project estimate. SEER-SEM does make adjustments for PLATFORM by changing the knowledge base, but these adjustments do not appear to be by the same order of magnitude as SASET or PRICE-S.

The selection of the development mode is another contributing factor explaining REVIC's lower estimates for CSCIs 1 and 3. The reader should recall each development uses a different development effort equation. Specifically, the Ada Development mode, used in CSCIs 1 and 3, has an exponent less than 1 where effort increases at a decreasing rate as effort increases. This is significant because the Ada development mode estimates less effort than the embedded or semidetached development modes (for CSCIs greater than 20,000 SLOC). Conversely, the embedded development mode equation used for CSCI2 has an exponent of 1.2. The different equations and exponents may explain why REVIC's estimate for CSCI2 are closer to the estimates calculated by the other models.

Unanticipated Results. It was anticipated that SEER-SEM would estimate more development effort than the other models since it is based on the minimum development time concept. However, this assumption was not accurate for the baseline test case. SEER-SEM calculated the second lowest project estimate

for the four models. Although SEER-SEM's estimates were always higher than REVIC's estimates, we cannot explain why it had lower estimates than PRICE-S or SASET for CSCIs 1 and 3. There may not be as much difference between SEER-SEM's minimum development time schedule and the other models' optimal schedule as originally believed. (See Chapter V, Conclusions and Recommendations, for more details regarding this situation.)

Conclusion. Recognizing the limitations of this simple baseline test case, several underlying reasons for models differences were identified and the researchers began to appreciate some of the difficulties associated with model normalization. One problem is that the models require different inputs and, outside of SLOC, they have different cost drivers and equations. For example, REVIC's development mode equations are effort drivers, but SASET does not have development mode equations. This makes it very difficult to adequately quantify the differences between the models.

The proprietary nature of SEER-SEM and PRICE-S also contribute to difficulties in comparing models. Although many of PRICE-S and SEER-SEM equations are published, many other equations are proprietary. Consequently, it is difficult to fully understand why these models provide different estimates than SASET and REVIC (whose equations are non-proprietary).

Project complexity also impacted the baseline test case. As project complexity and realism increased, it became very difficult identifying equivalent input values for each model. The initial baseline test case developed for this project was much more complex; however, it had to be simplified due to the incredible number of differences between the models. As a result, a very simple baseline case was used to avoid biasing the test case toward one of the models and allow the researchers to input logical and consistent values for each model.

Another difficulty with normalization is identifying an appropriate reference model. In other words, what model do you plan to normalize the model result to? Should the analyst identify SEER-SEM as the reference model and normalize all the other models to it or should REVIC be the reference model?

Furthermore, assuming each model had equivalent cost/effort drivers and identical parameters values, the models would provide different estimates since they were calibrated on different data bases. For example, the data base used to calibrate REVIC was significantly different than that used by PRICE-S. Likewise, SASET's data base bears little semblance to the data base used by Dr. Jensen to develop the initial SEER-SEM equations.

This does not imply it is useless to try to understand why models give different estimates. On the contrary, it can be helpful to understand these differences if more than one model is being used to estimate the same project. However, based on the effort expended during this baseline test case, it became obvious that model normalization is an exercise in futility for the average software cost model user. It is the researchers' opinion that the differences in definitions for model inputs, internal equations, and key assumptions make it nearly impossible to normalize the models (without indiscriminately adjusting model inputs to get equivalent estimates).

Ultimately, we feel users should become extremely familiar with one or two models. As the user becomes more experienced with the models, he or she should gain a great deal of insight in the capabilities of the model, the appropriate value for inputs, and proper calibration settings based on the project being estimated. This approach is more logical and defensible than running four different models and selecting the model with the desired cross-check estimate.

V. Conclusions and Recommendations

Overview

The purpose of this research effort was to develop a consolidated document which highlights the differences in definitions, assumptions, and methodologies used by the REVIC, SASET, PRICE-S, and SEER-SEM cost models and examines the impact of these differences on the resulting estimates. Conclusions regarding this effort are addressed in context of the three research objectives outlined in Chapter I, Introduction. Although many differences between the models were identified, this document does not cover all software development issues and is not a comprehensive source. Therefore, several recommendations for additional research are presented.

Conclusions

Three central research objectives guided this research effort. The conclusions will be addressed in context of these objectives.

Research Objective #1. What differences exist between the cost models? Differences between the models exist at nearly every level. At the onset of this project, the researchers did not realize the underlying equations and assumptions of the models were so diverse. However, as the research effort progressed, these differences became more and more evident.

Each model uses distinctly different equations for estimating development effort. In the case of REVIC, different equations are used within the same model depending on which development mode is selected. Additionally, the basic development concept (minimum development time versus minimum development effort) varied between the models.

Although the model developers' definitions for source lines of code were fairly consistent, each model has its own set of input parameters for describing the software development environment. The researchers determined it was nearly impossible to do a one-to-one correlation between model inputs and settled on a broad categorization of key model inputs.

The models also used different methodologies when estimating multiple CSCI projects and the impact of schedule compression and extensions. For example, SEER-SEM assumes total development effort decreases if the project is stretched out; however, the other models assume total development effort increases if the project is stretched out.

Lastly, the data bases used by each model were significantly different. Therefore, even if the models had the same underlying assumptions and equations, the estimates would vary due to differences in data base size, composition, and project attributes.

Research Objective #2. How do these differences impact the resulting cost estimates? Unfortunately, this question could not be answered in quantitative terms. For example, REVIC did not include a Systems Requirements Analysis/Design development phase. It was clearly beyond the researchers' abilities to quantify the dollar impact of this omission. Likewise, SASET has many more project attributes than the other models; however, the additional parameters resulted in only minor increases to the model estimate. As a result, differences were identified in relative terms rather than quantitative terms where possible.

Research Objective #3. To what degree can we explain and adjust for differences between cost models? The researchers found it was not particularly difficult to identify and discuss differences between the models. However, due to

the complexity associated with realistic development scenarios coupled with the different equations and underlying model assumptions, it is the researchers' opinion that model normalization is virtually impossible. The simple baseline test case supports this assertion. Of course, the user could systematically adjust the various model inputs to generate equivalent estimates. Yet, this approach defies logic since the user should have initially entered model inputs he or she deemed appropriate for the development effort.

Recommendations

This research effort did not address all possible facets of software cost estimating and several areas require additional research. First, one model developer questioned if many commonly used attributes are still applicable due to recent improvements in computer hardware (29). For example, additional research may determine that computer memory, utilization rates, or other factors are no longer constraining factors for software development projects.

Second, it would be interesting to perform a series of sensitivity analyses to determine how SEER-SEM's minimum development time solution compares to the minimum development effort solutions calculated by the other models. Although SEER-SEM's estimates were expected to be higher than the other models, the results of the baseline test case indicate this is not always true.

Third, additional research regarding the composition of each model's data base would also be beneficial. This research effort provided only a brief overview of the data bases, yet significant differences were highlighted. Identifying the estimating level (project versus CSCI), programming language, development contractor, and other distinctive characteristics could provide

additional insight into why one model estimates a particular class of software more accurately than another.

Fourth, this effort focused primarily on how models estimate effort. Another similar study should be undertaken to determine how the model's estimate schedules. For example, SASET generally estimated shorter schedules than REVIC even when it generated higher effort estimates. A specific issue to address would be the models' assumptions on overlapping phases and manpower staffing.

Fifth, additional research needs to be accomplished on the feasibility of normalization. Is it possible to objectively quantify the source of differences between model estimates? If so, how does one decide which model to normalize to and what are the benefits of normalization (does it result in a better estimate)?

In conclusion, this research effort identified many key similiarities and differences between four Air Force preferred software cost estimating models. It is the researchers' opinion that the differences in definitions for model inputs, internal equations, and key assumptions make it nearly impossible for the average model user to normalize the models. This does not imply it is useless to try to understand why models give different estimates. It is hoped that this effort resulted in a useful, consolidated document which explains the technical characteristics of the models and helps model users understand why the models produce different estimates.

Appendix A: Checklist Used to Examine Cost Models

1. What DoD-STD-2167A phases and activities are included in the estimate produced by each model?

a. Identify phases and activities specified by Mil Std 2167A.

b. Determine what DoD-STD-2167A phases and activities are included in the estimate produced by each model.

c. What are the basic assumptions or default values for the distribution of effort across the development phases?

d. Can the default allocation percentages be changed by the user?

2. What general development activities and specific cost elements are estimated?

a. What general development activities are included in the model estimates?

b. What specific cost elements are estimated by the model? What do they mean or represent?

c. If the model includes the cost of documentation, is that cost separately identified?

d. If the model includes the cost of program management, is the cost separately identified?

3. How does each model define a source line of code and how are language differences accounted for?

a. What is each model's definition for non-Ada source lines of code?

b. Do the models differentiate between Ada and non-Ada SLOC? If so, what are the models' definition of "Lines of Code" for Ada?

c. Do the models account for language differences?

4. Which model factors are key cost drivers? What estimating methodology was used to develop these factors?

a. Categorize important model factors based on personnel capabilities, development environment, project requirements, and target environment.

b. Identify key cost drivers used by each model to develop estimates (i.e which factors have the most significant impact on development effort?).

c. How were these factors developed? What estimating methodology was used? Linear regression or some other statistical method? Expert Judgment? Heuristics? Composite?

5. How does the model account for schedule compression and extensions?

a. Does the model allow schedule compression and extensions?

b. What, if any, penalties are assessed when the schedule is compressed or extended?

c. How sensitive are models to relatively small changes in schedules?

6. What implications does project size have on model output?

a. What is the basic assumption in the model concerning size and effort? Does effort increase at an increasing or decreasing rate as size increases?

b. What are the basic assumptions in the model concerning size and number of CSCIs, CSCs and CSUs in relation to the total effort and schedule? For example, if the total program is 100K SLOC, will breaking the project into multiple smaller units result in less or more total effort? Longer or shorter schedule?

e. What are the basic limitations of the model in terms of the size and number of CSCIs, CSCs and CSUs?

f. Is there a minimum or maximum size for the project or a CSCI? What is the recommended estimating range for the model?

7. Are there any distinctive characteristics about the data base(s) used by the various models?

a. How many projects were in the database used to develop the model?

b. Did it have any unique characteristics? If yes, how were these special characteristics normalized in developing the generic model?

c. How much of the database was military systems versus commercial systems? Embedded systems versus MIS systems?

d. What programming languages were included in the database (% each)?

e. What was the distribution of the records in the database by size (project level, CSCI, CSC, CSU)?

Appendix B: Model Inputs for Baseline Test Case

See subsequent pages for model inputs and results for the baseline test case.

MODEL INPUTS AND RESULTS FOR REVIC

REVIC MODEL PHASE DISTRIBUTION 08-03-1993 02:55:15

LOC to be developed is 50.0 KDSI (152 HRS/MM, $ 73.00 /HR)

PHASE & END REVIEW	EFFORT (mm)	SCHEDULE (months)	FSP (people)	COSTS
S/W RQMTS ENG (SRR)	37.2	8.2	4.5	413,319
PRELIM. DESIGN (PDR)*	71.4	10.7	6.7	792,194
CRITICAL DESIGN (CDR)*	90.0	6.9	13.1	998,854
CODE & DEBUG (TRR)*	68.3	4.1	16.6	757,751
INTEGRATE & TEST (FQT)*	80.7	5.8	14.0	895,524
DEV TEST & INT (DT&E)	68.3	7.7	8.9	757,751
TOTALS	416.0	43.4		$ 4,615,394

Total Productivity = 161.1 (246.8 programmers only) loc/mm
Environmental Modifier = 1.150 with a NM schedule
Total Direct Labor Hours = 63,225
ADA Software Development Mode
* - Items are included in Total Productivity calculation

REVIC MODEL ACTIVITY DISTRIBUTION

ACTIVITY	S/W RQMTS ENG (SRR)	PRELIM DSGN (PDR)	PROGR. (CDR & TRR)	I & T (FQT)
RQMTS ANALYSIS	17.13	7.14	4.75	1.61
PRODUCT DESIGN	5.21	29.09	9.50	3.23
PROGRAMMING	2.23	8.57	87.07	32.28
TEST PLANNING	1.49	4.28	9.50	3.23
VERIFY & VALIDATE	2.98	5.71	15.83	20.18
PROJECT OFFICE	4.47	7.85	11.08	6.46
CM/QA	1.49	2.14	11.08	7.26
MANUALS	2.23	5.71	9.50	6.46

NOTES: 1.0 MM = 152 HOURS
THE PROGRAMMING PHASE INCLUDES BOTH CRITICAL DESIGN AND SOFTWARE CODE & DEBUG.

REVIC MODEL RESULTS FOR

	-3 SIGMA	NOMINAL	+3 SIGMA
KDSI	50.0	50.0	50.0
MANMONTHS	416.0	416.0	416.0
SCHEDULE	43.4	43.4	43.4
TOTAL HOURS	63,225	63,225	63,225
TOTAL COSTS	$ 4,615,394	$ 4,615,394	$ 4,615,394

STANDARD DEVIATION = 0.000 KDSI

Environmental Factors for 08-03-1993 02:55:29

ENVIRONMENTAL FACTOR	RATING	VALUE	ENVIRONMENTAL FACTOR	RATING	VALUE
Analyst Capability	NM	1.00	Product Reliability	HI	1.15
Programmer Capability	NM	1.00	Data Base Size	NM	1.00
Applications Experience	NM	1.00	Product Complexity	NM	1.00
Virtual Machine Experience	NM	1.00	Required Reuse	NM	1.00
Prog. Language Experience	NM	1.00	Modern Programming Practices	NM	1.00
Execution Time Constraint	NM	1.00	Use Of S/W Tools	NM	1.00
Main Storage Constraint	NM	1.00	Required Security	UN	1.00
Virt. Machine Volatility	NM	1.00	Mgmt Reserve For Risk	VL	1.00
Computer Turnaround Time	NM	1.00	Required Schedule	NM	1.00
Requirements Volatility	NM	1.00	Software Development Mode	ADA	1.00

The environmental modifier is 1.150

Equivalent SEER Ctb = 5,707 Cte = 4,963

Version 9.1 - 23 November 1991

REVIC MODEL RESULTS FOR 08-03-1993 02:55:34
CDRL INITIAL PAGE ESTIMATES FOR DOD-STD-2167A/2168

SSDD, SRS, STD, AND STR PAGES = 671 ea.
IRS PAGES = 343
SDP PAGES = 175
SDD (PRELIMINARY) PAGES = 1666
SDD PAGES (FINAL, WITHOUT LISTINGS) = 5000
STP, CSOM PAGES = 135 ea.
IDD PAGES = 676
SPS, SPM PAGES = 72 ea.
SUM, CRISD PAGES = 67 ea.
FSM PAGES = 53
VDD PAGES APPROXIMATELY 10 PER FORMAL MEDIA RELEASE.

S/W DEVELOPMENT FOLDER PAGES NOT INCLUDED HERE.

Page counts are approximate for each document. See DOD-STD-2167A for an explanation of the acronyms and a description of their content.

INDEX #	NAME	LE	MP	HE
1		20000	20000	20000
2		30000	30000	30000

INDEX #	NAME	ADSI	DM	CM	IM

REVIC MODEL PHASE DISTRIBUTION 08-03-1993 02:59:25

LOC to be developed is 80.0 KDSI (152 HRS/MM, $ 73.00 /HR)

PHASE & END REVIEW		EFFORT (mm)	SCHEDULE (months)	FSP (people)	COSTS
S/W RQMTS ENG	(SRR)	71.8	10.2	7.1	796,280
PRELIM. DESIGN	(PDR)*	[illegible]	13.2	10.4	1,526,204
CRITICAL DESIGN	(CDR)*	173.4	8.5	20.5	1,924,344
CODE & DEBUG	(TRR)*	131.6	5.1	25.9	1,459,848
INTEGRATE & TEST	(FQT)*	155.5	7.1	21.9	1,725,274
DEV TEST & INT	(DT&E)	167.4	9.5	17.7	1,857,988
TOTALS		837.2	53.5		$ 9,289,938

Total Productivity = 133.8 (205.0 programmers only) loc/mm
Environmental Modifier = 0.940 with a NM schedule
Total Direct Labor Hours = 127,259
EMBEDDED Software Development Mode
* - Items are included in Total Productivity calculation

REVIC MODEL ACTIVITY DISTRIBUTION

ACTIVITY	S/W RQMTS ENG (SRR)	PRELIM DSGN (PDR)	PROGR. (CDR & TRR)	I & T (FQT)
RQMTS ANALYSIS	33.01	13.75	9.15	3.11
PRODUCT DESIGN	10.05	57.77	18.30	6.22
PROGRAMMING	4.31	16.51	167.75	62.19
TEST PLANNING	2.87	8.25	18.30	6.22
VERIFY & VALIDATE	5.74	11.00	30.50	38.87
PROJECT OFFICE	8.61	15.13	21.35	12.44
CM/QA	2.87	4.13	21.35	13.99
MANUALS	4.31	11.00	18.30	12.44

NOTES: 1.0 MM = 152 HOURS
THE PROGRAMMING PHASE INCLUDES BOTH CRITICAL DESIGN
AND SOFTWARE CODE & DEBUG.

REVIC MODEL RESULTS FOR

	-1 SIGMA	NOMINAL	+1 SIGMA
KDSI	80.0	80.0	80.0
MANMONTHS	837.2	837.2	837.2
SCHEDULE	53.5	53.5	53.5
TOTAL HOURS	127,259	127,259	127,259
TOTAL COSTS	$ 9,289,938	$ 9,289,938	$ 9,289,938

STANDARD DEVIATION = 0.000 KDSI

Environmental Factors for 08-03-1993 02:59:38

ENVIRONMENTAL FACTOR	RATING	VALUE	ENVIRONMENTAL FACTOR	RATING	VALUE
Analyst Capability	NM	1.00	Product Reliability	HI	1.15
Programmer Capability	HI	0.86	Data Base Size	NM	1.00
Applications Experience	NM	1.00	Product Complexity	NM	1.00
Virtual Machine Experience	NM	1.00	Required Reuse	NM	1.00
Prog. Language Experience	HI	0.95	Modern Programming Practices	NM	1.00
Execution Time Constraint	NM	1.00	Use Of S/W Tools	NM	1.00
Main Storage Constraint	NM	1.00	Required Security	UN	1.00
Virt. Machine Volatility	NM	1.00	Mgmt Reserve For Risk	VL	1.00
Computer Turnaround Time	NM	1.00	Required Schedule	NM	1.00
Requirements Volatility	NM	1.00	Software Development Mode	EB	1.00

The environmental modifier is 0.940

Equivalent SEER Ctb = 6,383 Cte = 5,843

Version 9.1 - 23 November 1991

REVIC MODEL RESULTS FOR 08-03-1993 02:59:43
CDRL INITIAL PAGE ESTIMATES FOR DOD-STD-2167A/2168

SSDD, SRS, STD, AND STR PAGES = 1071 ea.
IRS PAGES = 543
SDP PAGES = 250
SDD (PRELIMINARY) PAGES = 2666
SDD PAGES (FINAL, WITHOUT LISTINGS) = 8000
STP, CSOM PAGES = 210 ea.
IDD PAGES = 1076
SP:, SPM PAGES = 110 ea.
SUM, CRISD PAGES = 105 ea.
FSM PAGES = 83
VDD PAGES APPROXIMATELY 10 PER FORMAL MEDIA RELEASE.

S/W DEVELOPMENT FOLDER PAGES NOT INCLUDED HERE.

Page counts are approximate for each document. See DOD-STD-2167A for an explanation of the acronyms and a description of their content.

INDEX #	NAME	LE	MP	HE
1		80000	80000	80000

INDEX #	NAME	ADSI	DM	CM	IM

REVIC MODEL PHASE DISTRIBUTION 08-03-1993 03:01:36

LOC to be developed is 45.0 KDSI (152 HRS/MM, $ 73.00 /HR)

PHASE & END REVIEW		EFFORT (mm)	SCHEDULE (months)	FSP (people)	COSTS
S/W RQMTS ENG	(SRR)	33.7	8.0	4.2	374,307
PRELIM. DESIGN	(PDR)*	64.7	10.4	6.2	717,421
CRITICAL DESIGN	(CDR)*	81.5	6.6	12.3	904,574
CODE & DEBUG	(TRR)*	61.8	4.0	15.5	686,22?
INTEGRATE & TEST	(FQT)*	73.1	5.6	13.1	810,9??
DEV TEST & INT	(DT&E)	61.8	7.4	8.3	686,229
TOTALS		376.7	42.0		$ 4,179,757

Total Productivity = 160.1 (245.3 programmers only) loc/mm
Environmental Modifier = 1.150 with a NM schedule
Total Direct Labor Hours = 57,257
ADA Software Development Mode
* - Items are included in Total Productivity calculation

REVIC MODEL ACTIVITY DISTRIBUTION

ACTIVITY	S/W RQMTS ENG (SRR)	PRELIM DSGN (PDR)	PROGR. (CDR & TRR)	I & T (FQT)
RQMTS ANALYSIS	15.52	6.47	4.30	1.46
PRODUCT DESIGN	4.72	27.16	8.60	2.92
PROGRAMMING	2.02	7.76	78.85	29.24
TEST PLANNING	1.35	3.88	8.60	2.92
VERIFY & VALIDATE	2.70	5.17	14.34	18.27
PROJECT OFFICE	4.05	7.11	10.04	5.85
CM/QA	1.35	1.94	10.04	6.58
MANUALS	2.02	5.17	8.60	5.85

NOTES: 1.0 MM = 152 HOURS
THE PROGRAMMING PHASE INCLUDES BOTH CRITICAL DESIGN
AND SOFTWARE CODE & DEBUG.

REVIC MODEL RESULTS FOR

	-3 SIGMA	NOMINAL	+3 SIGMA
KDSI	45.0	45.0	45.0
MANMONTHS	376.7	376.7	376.7
SCHEDULE	42.0	42.0	42.0
TOTAL HOURS	57,257	57,257	57,257
TOTAL COSTS	$ 4,179,756	$ 4,179,756	$ 4,179,756

STANDARD DEVIATION = 0.000 KDSI

Environmental Factors for 08-03-1993 03:01:50

ENVIRONMENTAL FACTOR	RATING	VALUE	ENVIRONMENTAL FACTOR	RATING	VALUE
Analyst Capability	NM	1.00	Product Reliability	HI	1.15
Programmer Capability	NM	1.00	Data Base Size	NM	1.00
Applications Experience	NM	1.00	Product Complexity	NM	1.00
Virtual Machine Experience	NM	1.00	Required Reuse	NM	1.00
Prog. Language Experience	NM	1.00	Modern Programming Practices	NM	1.00
Execution Time Constraint	NM	1.00	Use Of S/W Tools	NM	1.00
Main Storage Constraint	NM	1.00	Required Security	UN	1.00
Virt. Machine Volatility	NM	1.00	Mgmt Reserve For Risk	VL	1.00
Computer Turnaround Time	NM	1.00	Required Schedule	NM	1.00
Requirements Volatility	NM	1.00	Software Development Mode	ADA	1.00

The environmental modifier is 1.150

Equivalent SEER Ctb = 5,707 Cte = 4,963

Version 9.1 - 23 November 1991

REVIC MODEL RESULTS FOR 08-03-1993 03:01:54
CDRL INITIAL PAGE ESTIMATES FOR DOD-STD-2167A/2168

SSOD, SRS, STP, AND STR PAGES = 605 ea.
IRS PAGES = 310
SDP PAGES = 162
SDD (PRELIMINARY) PAGES = 1500
SDD PAGES (FINAL, WITHOUT LISTINGS) = 4500
STP, CSCM PAGES = 122 ea.
IDD PAGES = 610
SPS, SPM PAGES = 66 ea.
SUM, CRISD PAGES = 61 ea.
FSM PAGES = 48
VDD PAGES APPROXIMATELY 10 PER FORMAL MEDIA RELEASE.

S/W DEVELOPMENT FOLDER PAGES NOT INCLUDED HERE.

Page counts are approximate for each document. See DOD-STD-2167A for an explanation of the acronyms and a description of their content.

INDEX #	NAME	LE	MP	HE
1		45000	45000	45000

INDEX #	NAME	ADSI	DM	CM	IM

MODEL INPUTS AND RESULTS FOR SASET

****** System Environment Distribution ******

Title	Budget	Schedule	Value
Class of Software	1.800	1.450	Avionics
Hardware System Type	0.950	0.950	Centralized
Pct of Memory Utilized	0.950	0.990	50 %
S/W Configuration Items	1.050	1.050	3
Development Locations	1.000	1.000	1
Customer Locations	1.000	1.000	1
Dev. Workstation Types	1.000	1.000	1
Primary Software Language	0.875	1.000	FORTRAN,Pascal,Jovial,C
Pct of Micro-Code	1.000	1.000	0 %
Lifecycle Choice	1.000	1.000	DoD-Std-2167A
Software Budget Multiplier	0.82917		
Software Schedule Multiplier	0.98752		
Budget Data Factor	15.000		
Schedule Data Factor	12.000		

SOURCE FILE: C.\SASET30\BASELINE.FMS
3 Aug 1993 2:44:37 AM

Project : baseline.sas	S/W TYPE	PERCENT N/M/R	DATA STMT	SLOC
PROJECT: baseline.sas	-	-	0	175000
CSCI: csci1	-	-	0	50000
CSC: CSC2	-	-	0	30000
CSC: CSC1	-	-	0	20000
CSCI: CSCI2	-	-	0	80000
CSC: CSCI2	-	-	0	80000
CSCI: CSCI3	-	-	0	45000
CSC: CSCI3	-	-	0	45000

Summary of Staff Hours / Staff Months / Hours per Line Code
based on Conditioned HOL Equivalents by Software Type

Software Type	Staff Hours	Staff Months	Cond. HOL Equivalents	Hours Per LOC
Systems	0.00	0.0	0	0.00000
Applications	128398.22	844.7	50000	2.56796
Support	0.00	0.0	0	0.00000
Budget =	128398.22	844.7	50000	2.56796
Data Statements	0.00	0.0	0	
Systems Reqts	20140.90	132.5		
Systems Test	19301.69	127.0		
Total Budget =	167840.82	1104.2		

-> Hours per LOC is 3.35682 <-

****** System Attributes Distribution ******

System Complexity Title	Budget	Schedule	Complexity
System Requirements	1.000	1.000	Average
Software Requirements	1.000	1.000	Average
Software Documentation	1.020	1.010	Complex
Travel Requirements	1.000	1.000	Average
Man Interaction	1.020	1.010	Complex
Timing and Criticality	1.000	1.000	Average
Software Testability	1.000	1.000	Average
Hardware Constraints	1.000	1.000	Average
Hardware Experience	1.000	1.000	Average
Software Experience	1.000	1.000	Average
Software Interfaces	1.000	1.000	Average
Development Facilities	1.000	1.000	Average
Development vs Host Sys	1.000	1.000	Average
Technology Impacts	1.000	1.000	Average
COTS Software	1.000	1.000	Average
Development Team	1.000	1.000	Average
Embedded Development Sys	1.000	1.000	Average
Software Development Tools	1.000	1.000	Average
Personnel Resources	1.000	1.000	Average
Programming Language	1.000	1.000	Average
Software Systems Budget Multiplier		1.04040	
Software Systems Schedule Multiplier		1.02010	

****** System Attributes Distribution ******

Integration Complexity Title	Factor	Cmplx	Product	Complexity
S/W Language Complexity	9.00	2.00	18.00	Average
Modularity of Software	5.00	2.00	10.00	Average
S/W Timing & Criticality	5.00	2.00	10.00	Average
Number of CSCI Interfaces	7.00	2.00	14.00	Average
Software Documentation	7.00	3.00	21.00	Complex
Development Facilities	4.00	2.00	8.00	Average
Software Interfaces	6.00	2.00	12.00	Average
Testing Complexity	8.00	2.00	16.00	Average
Development Complexity	7.00	2.00	14.00	Average
Integration Experience	6.00	2.00	12.00	Average
Integ. Development Tools	6.00	2.00	12.00	Average
Schedule Constraints	8.00	2.00	16.00	Average
Budget Increase % = 5.0 %		Budget Value =	163.00	

Summary of Staff Hours / Staff Months / Hours per Line Code
based on Conditioned HOL Equivalents by Software Type

Software Type	Staff Hours	Staff Months	Cond. HOL Equivalents	Hours Per LOC
Systems	0.00	0.0	0	0.00000
Applications	63410.69	417.2	26667	2.37790
Support	0.00	0.0	0	0.00000
Budget =	63410.69	417.2	26667	2.37790
Data Statements	0.00	0.0	0	
Systems Reqts	9946.78	65.4		
Systems Test	9532.33	62.7		
Total Budget =	82889.79	545.3		

-> Hours per LOC is 3.10837 <-

****** System Attributes Distribution ******

System Complexity Title	Budget	Schedule	Complexity
System Requirements	1.000	1.000	Average
Software Requirements	1.000	1.000	Average
Software Documentation	1.020	1.010	Complex
Travel Requirements	1.000	1.000	Average
Man Interaction	1.020	1.010	Complex
Timing and Criticality	1.000	1.000	Average
Software Testability	1.000	1.000	Average
Hardware Constraints	1.000	1.000	Average
Hardware Experience	1.000	1.000	Average
Software Experience	0.850	0.990	Simple
Software Interfaces	1.000	1.000	Average
Development Facilities	1.000	1.000	Average
Development vs Host Sys	1.000	1.000	Average
Technology Impacts	1.000	1.000	Average
COTS Software	1.000	1.000	Average
Development Team	1.000	1.000	Average
Embedded Development Sys	1.000	1.000	Average
Software Development Tools	1.000	1.000	Average
Personnel Resources	1.000	1.000	Average
Programming Language	1.020	1.010	Complex
Software Systems Budget Multiplier		0.90203	
Software Systems Schedule Multiplier		1.02000	

****** System Attributes Distribution ******

Integration Complexity Title	Factor	Cmplx	Product	Complexity
S/W Language Complexity	9.00	3.00	27.00	Complex
Modularity of Software	5.00	2.00	10.00	Average
S/W Timing & Criticality	5.00	2.00	10.00	Average
Number of CSCI Interfaces	7.00	2.00	14.00	Average
Software Documentation	7.00	3.00	21.00	Complex
Development Facilities	4.00	2.00	8.00	Average
Software Interfaces	6.00	2.00	12.00	Average
Testing Complexity	8.00	2.00	16.00	Average
Development Complexity	7.00	2.00	14.00	Average
Integration Experience	6.00	2.00	12.00	Average
Integ. Development Tools	6.00	2.00	12.00	Average
Schedule Constraints	8.00	2.00	16.00	Average
Budget Increase % = 5.0 %		Budget Value =	172.00	

Summary of Staff Hours / Staff Months / Hours per Line Code
based on Conditioned HOL Equivalents by Software Type

Software Type	Staff Hours	Staff Months	Cond. HOL Equivalents	Hours Per LOC
Systems	0.00	0.0	0	0.00000
Applications	115558.40	760.3	45000	2.56796
Support	0.00	0.0	0	0.00000
Budget =	115558.40	760.3	45000	2.56796
Data Statements	0.00	0.0	0	
Systems Reqts	18126.81	119.3		
Systems Test	17371.52	114.3		
Total Budget =	151056.73	993.8		

-> Hours per LOC is 3.35682 <-

****** System Attributes Distribution ******

System Complexity Title	Budget	Schedule	Complexity
System Requirements	1.000	1.000	Average
Software Requirements	1.000	1.000	Average
Software Documentation	1.020	1.010	Complex
Travel Requirements	1.000	1.000	Average
Man Interaction	1.020	1.010	Complex
Timing and Criticality	1.000	1.000	Average
Software Testability	1.000	1.000	Average
Hardware Constraints	1.000	1.000	Average
Hardware Experience	1.000	1.000	Average
Software Experience	1.000	1.000	Average
Software Interfaces	1.000	1.000	Average
Development Facilities	1.000	1.000	Average
Development vs Host Sys	1.000	1.000	Average
Technology Impacts	1.000	1.000	Average
COTS Software	1.000	1.000	Average
Development Team	1.000	1.000	Average
Embedded Development Sys	1.000	1.000	Average
Software Development Tools	1.000	1.000	Average
Personnel Resources	1.000	1.000	Average
Programming Language	1.000	1.000	Average
Software Systems Budget Multiplier		1.04040	
Software Systems Schedule Multiplier		1.02010	

****** System Attributes Distribution ******

Integration Complexity Title	Factor	Cmplx	Product	Complexity
S/W Language Complexity	9.00	2.00	18.00	Average
Modularity of Softwar	5.00	2.00	10.00	Average
S/W Timing & Criticality	5.00	2.00	10.00	Average
Number of CSCI Interfaces	7.00	2.00	14.00	Average
Software Documentation	7.00	3.00	21.00	Complex
Development Facilities	4.00	2.00	8.00	Average
Software Interfaces	6.00	2.00	12.00	Average
Testing Complexity	8.00	2.00	16.00	Average
Development Complexity	7.00	2.00	14.00	Average
Integration Experience	6.00	2.00	12.00	Average
Integ. Development Tools	6.00	2.00	12.00	Average
Schedule Constraints	8.00	2.00	16.00	Average

Budget Increase % = 5.0 % Budget Value = 163.00

MODEL INPUTS AND RESULTS FOR PRICE-S

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 3:59 PM Project : Project1
392148

CSCI 1 Devt. Item w/comps

ITEM DESCRIPTORS
Platform 1.80 Mgmt Complexity 1.00 External Integ 0.50

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Spec. Review	894
Pre. Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

COMPONENT 1 titled: CSC 1

DESCRIPTORS

Internal Integ	0.50	External Integ	0.50
Utilization Fraction	0.50		

SCHEDULE

Software Spec. Review	894	Pre. Design Review	0
Critical Design Review	0	Test Readiness Review	0
Functional Config Audit	0		

LANGUAGE 1 DESCRIPTORS

Language Ada		Source Code	30000	Non-executable SLOC	0.00
Complexity 1	1.00	Complexity 2	1.00	Productivity Factor	5.00
Application	5.50	New Design	1.00	New Code	1.00

Application Categories		Mix	New Design	New Code
User Defined (APPL =	5.50)	0.00	0.00	0.00
DATA S/R		0.00	1.00	1.00
Online Comm		0.00	1.00	1.00
Realtime C&C		0.00	0.00	0.00
Interactive		0.00	1.00	1.00
Mathematical		0.00	1.00	1.00
String Manip		0.00	1.00	1.00
Operating Systems		0.00	1.00	1.00

COMPONENT 2 titled: CSC 2

DESCRIPTORS

Internal Integ	0.50	External Integ	0.50
Utilization Fraction	0.50		

SCHEDULE

Software Spec. Review	894	Pre. Design Review	0
Critical Design Review	0	Test Readiness Review	0
Functional Config Audit	0		

LANGUAGE 1 DESCRIPTORS

Language Ada		Source Code	20000	Non-executable SLOC	0.00
Complexity 1	1.00	Complexity 2	1.00	Productivity Factor	5.00
Application	5.50	New Design	1.00	New Code	1.00

Application Categories	Mix	New Design	New Code

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 3/59 PM Project : Project1
392148

CSCI 1 Devt. Item w/comps

Costs in Person Months

	Design	Pgming	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	27.0	0.0	4.7	12.6	1.2	1.2	46.5
Sys/SW Reqt	33.7	0.0	5.8	15.7	1.5	1.5	58.1
SW Requirement	41.7	0.0	13.9	52.7	6.9	6.9	122.1
Prelim Design	76.5	24.7	21.6	50.9	11.1	11.1	196.0
Detail Design	114.7	37.1	32.5	76.3	16.7	16.7	294.0
Code/Test	37.4	136.7	23.3	28.5	30.5	30.5	287.0
CSCI Test	70.0	48.5	35.1	38.1	38.3	38.3	268.1
System Test	21.0	26.3	4.2	12.6	13.7	27.3	105.1
Oper TE	13.1	7.8	5.2	7.8	8.9	9.4	52.3
TOTAL	434.9	281.2	146.3	295.2	128.8	142.9	1429.5

SCHEDULE INFORMATION

Concept Start	Mar 93*		TRR	Sep 96*	(11.4)
SRR	Jul 93*	(4.9)	FCA	Apr 97*	(7.0)
SDR	Nov 93*	(3.2)	PCA	Jun 97*	(2.0)
SSR	Aug 94	(9.5)	FQR	Aug 97*	(2.0)
PDR	Feb 95*	(6.4)	OTE	Jan 98*	(5.1)
CDR	Oct 95*	(7.2)			

SUPPLEMENTAL INFORMATION

Source Lines of Code	50000
Source Lines of Code/ Person Months	47.83

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4:02 PM Project : Project1
392148

CSCI 1 Devt. Item w/comps

Costs in Person Months

SENSITIVITY DATA
(PROFAC - COMPLEXITY)

COMPLEXITY CHANGE

PROFAC CHANGE	- 0.50		0.0		+ 0.50	
+ 0.50	COST	825.2	COST	1300.9	COST	1783.7
	MONTHS	26.4	MONTHS	58.0	MONTHS	92.1
0.00	COST	909.4	COST	1429.5	COST	1961.7
	MONTHS	26.7	MONTHS	58.7	MONTHS	93.5
- 0.50	COST	1012.6	COST	1585.7	COST	2171.0
	MONTHS	27.1	MONTHS	59.4	MONTHS	94.7

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/11/93 TIME 4/06 PM Project : Project1
392148

CSCI 1 Devt. Item w/comps

Costs in Person Months

SENSITIVITY DATA
(APPLICATION - SIZE)

SIZE CHANGE

APPL CHANGE		- 10.0%	0.0	+ 10.0%
- 0.10	COST	1251.8	1404.0	1557.6
	MONTHS	56.0	58.2	60.3
0.00	COST	1274.4	1429.5	1585.9
	MONTHS	56.4	58.7	60.8
+ 0.10	COST	1297.1	1454.9	1614.2
	MONTHS	56.9	59.1	61.3

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4/06 PM Project : Project1
392148

CSCI 2 Development Item

ITEM DESCRIPTORS

Platform	1.80	Mgmt Complexity	1.00	Cost	0.00
Internal Integ	0.50	External Integ	0.50	Utilization	0.50

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Spec. Review	894
Pre. Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language Assembly		Source Code	80000	Non-executable SLOC	0.00
Complexity 1	0.80	Complexity 2	1.00	Productivity Factor	5.00
Application	5.50	New Design	1.00	New Code	1.00

Application Categories		Mix	New Design	New Code
User Defined (APPL =	6.42)	0.00	0.00	0.00
DATA S/R		0.00	0.00	0.00
Online Comm		0.00	0.00	0.00
Realtime C&C		0.00	0.00	0.00
Interactive		0.00	0.00	0.00
Mathematical		0.00	0.00	0.00
Strin Manip		0.00	0.00	0.00
Operating Systems		0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4/06 PM Project : Project1
392148

CSCI 2 Development Item

Costs in Person Months

	Design	Pgming	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	10.3	0.0	1.8	4.8	0.4	0.4	17.8
Sys/SW Reqt	12.9	0.0	2.2	6.0	0.6	0.6	22.2
SW Requirement	16.8	0.0	4.8	21.0	2.4	2.4	47.4
Prelim Design	25.7	9.3	6.6	16.9	3.3	3.3	65.0
Detail Design	38.6	13.9	9.9	25.3	4.9	4.9	97.5
Code/Test	10.1	48.4	6.7	8.4	8.9	8.9	91.4
CSCI Test	32.1	21.4	15.5	17.8	16.1	16.1	118.8
System Test	8.0	10.0	1.6	4.8	5.2	10.4	40.1
Oper TE	5.0	3.0	2.0	1.0	1.4	1.6	20.0
TOTAL	159.4	105.9	51.1	108.0	45.2	50.6	520.2

SCHEDULE INFORMATION

Concept Start	Nov 93*		TRR	Sep 95*	(6.0)
SRR	Dec 93*	(2.9)	FCA	Feb 96*	(4.5)
SDR	Feb 94*	(1.9)	PCA	Mar 96*	(1.4)
SSR	Aug 94	(5.6)	FQR	Apr 96*	(1.4)
PDR	Nov 94*	(3.4)	OTE	Aug 96*	(3.4)
CDR	Mar 95*	(3.7)			

SUPPLEMENTAL INFORMATION

Source Lines of Code	80000
Source Lines of Code/ Person Months	214.64

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4/07 PM Project : Project1
392148

CSCI 2 Development Item

Costs in Person Months

SENSITIVITY DATA
(PROFAC - COMPLEXITY)

COMPLEXITY CHANGE

PROFAC CHANGE	- 0.50		0.0		+ 0.50	
+ 0.50	COST	308.7	COST	575.2	COST	841.4
	MONTHS	11.3	MONTHS	34.5	MONTHS	60.5
0.00	COST	277.9	COST	520.2	COST	760.5
	MONTHS	11.2	MONTHS	34.1	MONTHS	59.7
- 0.50	COST	252.3	COST	474.4	COST	697.4
	MONTHS	11.0	MONTHS	33.7	MONTHS	59.0

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4/08 PM Project : Project1
392148

CSCI 2 Development Item

Costs in Person Months

SENSITIVITY DATA
(APPLICATION - SIZE)

SIZE CHANGE

APPL CHANGE	- 10.0%		0.0		+ 10.0%	
- 0.10	COST	455.8	COST	511.3	COST	567.3
	MONTHS	32.5	MONTHS	33.8	MONTHS	35.0
0.00	COST	463.7	COST	520.2	COST	577.2
	MONTHS	32.8	MONTHS	34.1	MONTHS	35.3
+ 0.10	COST	471.6	COST	529.1	COST	587.1
	MONTHS	33.0	MONTHS	34.4	MONTHS	35.6

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4/08 PM Project : Project1
392148

CSCI 3 Development Item

ITEM DESCRIPTORS

Platform	1.80	Mgmt Complexity	1.00	Cost	0.00
Internal Integ	0.50	External Integ	0.50	Utilization	0.50

ITEM SCHEDULE

System Concept Date	0	System Requirements Review	0
System Design Review	0	Software Spec. Review	894
Pre. Design Review	0	Critical Design Review	0
Test Readiness Review	0	Functional Config Audit	0
Physical Config Audit	0	Functional Qual Review	0
Oper Test & Evaluation	0		

LANGUAGE 1 DESCRIPTORS

Language Ada		Source Code	45000	Non-executable SLOC	0.00
Complexity 1	1.00	Complexity 2	1.00	Productivity Factor	5.00
Application	5.50	New Design	1.00	New Code	1.00

Application Categories		Mix	New Design	New Code
User Defined (APPL =	5.50)	1.00	1.00	1.00
DATA S/R		0.00	0.00	0.00
Online Comm		0.00	0.00	0.00
Realtime C&C		0.00	0.00	0.00
Interactive		0.00	0.00	0.00
Mathematical		0.00	0.00	0.00
String Manip		0.00	0.00	0.00
Operating Systems		0.00	0.00	0.00

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4/08 PM Project : Project1
392148

CSCI 3

Development Item

Costs in Person Months

	Design	Pgming	Data	S/PM	Q/A	Config	TOTAL
Sys Concept	24.5	0.0	4.2	11.4	1.1	1.1	42.3
Sys/SW Reqt	30.6	0.0	5.3	14.3	1.3	1.3	52.8
SW Requirement	37.3	0.0	12.3	47.1	6.1	6.1	108.9
Prelim Design	63.1	20.5	19.1	42.9	9.8	9.8	165.1
Detail Design	94.7	30.8	28.6	64.4	14.7	14.7	247.7
Code/Test	22.1	102.0	15.8	19.1	21.5	21.5	201.9
CSCI Test	63.3	44.0	14.1	35.6	36.6	36.6	250.1
System Test	19.1	23.9	3.8	11.5	12.4	24.8	95.5
Oper TE	11.9	7.1	4.8	7.1	8.1	8.6	47.5
TOTAL	366.6	228.2	127.9	253.3	111.4	124.3	1211.8

SCHEDULE INFORMATION

Concept Start	Apr 93*		TRR	Jun 96*	(9.2)
SRR	Aug 93*	(4.7)	FCA	Jan 97*	(6.7)
SDR	Nov 93*	(3.1)	PCA	Mar 97*	(2.0)
SSR	Aug 94	(9.1)	FQR	May 97*	(2.0)
PDR	Feb 95*	(6.2)	OTE	Oct 97*	(4.9)
CDR	Sep 95*	(6.9)			

SUPPLEMENTAL INFORMATION

Source Lines of Code	45000
Source Lines of Code/ Person Months	52.03

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4:09 PM Project : Project1
392148

CSCI 3 Development Item

Costs in Person Months

SENSITIVITY DATA
(PROFAC - COMPLEXITY)

COMPLEXITY CHANGE

PROFAC CHANGE	- 0.50		0.0		+ 0.50	
- 0.50	COST	819.1	COST	1343.2	COST	1878.9
	MONTHS	25.0	MONTHS	55.4	MONTHS	88.8
0.00	COST	736.4	COST	1211.8	COST	1698.9
	MONTHS	24.6	MONTHS	54.7	MONTHS	87.7
+ 0.50	COST	668.5	COST	1103.4	COST	1549.7
	MONTHS	24.3	MONTHS	54.1	MONTHS	86.7

--- PRICE SOFTWARE MODEL ---
Acquisition Mode

DATE Sat 7/31/93 TIME 4:11 PM Project : Project1
392148

CSCI 3 Development Item

Costs in Person Months

SENSITIVITY DATA
(APPLICATION - SIZE)

SIZE CHANGE

APPL CHANGE	- 10.0%		0.0		+ 10.0%	
- 0.10	COST	1060.5	COST	1190.1	COST	1320.9
	MONTHS	52.2	MONTHS	54.3	MONTHS	56.3
0.00	COST	1079.9	COST	1211.8	COST	1345.0
	MONTHS	52.6	MONTHS	54.7	MONTHS	56.7
+ 0.10	COST	1099.2	COST	1233.6	COST	1369.2
	MONTHS	53.0	MONTHS	55.1	MONTHS	57.1

MODEL INPUTS AND RESULTS FOR SEER-SEM

SEER-SEM (TM) Software Schedule, Cost & Risk Estimation Version 3.21
Project : Test Case 7/31/93
PROJECT : Test Case 4:20:22 PM

Quick Estimate

Effort Months	2,366.37
Base Year Cost	33,602.42K
Phases Included	REQ + FSI + STE

 Page 1

Activity

Activity	Schedule Months	Person Months	Person Hours	Cost
System Concept	0.00	0.00	0	0
Cumulative	0.00	0.00	0	0
System Requirements Design	0.00	33.60	5,107	477,058
Cumulative	0.00	33.60	5,107	477,058
S/W Requirements Analysis	0.00	98.59	14,986	1,400,019
Cumulative	0.00	132.19	20,093	1,877,078
Preliminary Design	0.00	203.43	30,921	2,888,640
Cumulative	0.00	335.61	51,013	4,765,717
Detailed Design	0.00	341.19	51,861	4,844,924
Cumulative	0.00	676.81	102,874	9,610,641
Code & CSU Test	0.00	534.22	81,201	7,585,881
Cumulative	0.00	1,211.02	184,075	17,196,522
CSC Integrate & Test	0.00	637.83	96,951	9,057,233
Cumulative	0.00	1,848.86	281,026	26,253,756
CSCI Test	0.00	73.34	11,148	1,041,475
Cumulative	0.00	1,922.20	292,174	27,295,230
System Integrate Thru OT&E	0.00	444.17	67,514	6,307,189
Cumulative	0.00	2,366.37	359,688	33,602,419
Maintenance / Op Support	0.00	0.00	0	0
Cumulative	0.00	2,366.37	359,688	33,602,419

SEER-SEM (TM) Software Schedule, Cost & Risk Estimation Version 3.21
Project : Test Case 7/31/93
CSCI : CSCI 1 4:20:38 PM

Quick Estimate

Schedule Months	41.29
Effort Months	719.41
Base Year Cost	10,215.59K
Constraint	MIN TIME
Phases Included	REQ + FSI + STE

Inputs

	Least	Likely	Most
+ EFFECTIVE LINES	50,000	50,000	50,000
- New Lines of Code	50,000	50,000	50,000
+ PRE-EXISTS, NOT DESIGNED FOR REUSE	0	0	0
- Pre-existing lines of code	0	0	0
- Lines to be deleted in pre-exsty	0	0	0
- Redesign required	0.00%	0.00%	0.00%
- Reimplementation required	0.00%	0.00%	0.00%
- Retest required	0.00%	0.00%	0.00%
+ PRE-EXISTS, DESIGNED FOR REUSE	0	0	0
- COMPLEXITY	VHi-	VHi	VHi+
+ PERSONNEL CAPABILITIES & EXPERIENCE	Low-	Nom	Hi-
- Analyst Capabilities	Low	Nom	Hi
- Analyst's Application Experience	Nom-	Nom	Nom+
- Programmer Capabilities	Low	Nom	Hi
- Programmer's Language Experience	Low-	Nom	Hi
- Host Development System Experience	Low-	Nom	Hi
- Target System Experience	Low-	Nom	Hi
- Practices & Methods Experience	Low-	Nom	Hi
+ DEVELOPMENT SUPPORT ENVIRONMENT	Low+	Nom	Nom
- Modern Development Practices Use	Nom-	Nom	Nom+
- Automated Tools Use	Low	Nom	Nom
- Logon thru Hardcopy Turnaround	VLo	Low+	Hi
- Terminal Response Time	Low-	Hi-	Hi
- Multiple Site Development	Nom	Nom	Nom
- Resource Dedication	Nom	Nom	Nom
- Resource and Support Location	Nom	Nom	Nom
- Host System Volatility	Nom	Nom	Nom
- Practices & Methods Volatility	Nom	Nom	Nom
+ PRODUCT DEVELOPMENT REQUIREMENTS	Nom	Nom	Hi
- Requirements Volatility (Change)	Nom	Nom	Hi
- Specification Level - Reliability	Nom	Nom	Hi
- Test Level	Nom	Nom	Hi
- Quality Assurance Level	Nom	Nom	Hi
- Rehost from Development to Target	Nom	Hi	VHi+
+ PRODUCT REUSABILITY REQUIREMENTS			
- Reusability Level Required	Nom	Nom	Nom
- Software Impacted by Reuse	0.00%	0.00%	0.00%
+ DEVELOPMENT ENVIRONMENT COMPLEXITY	Nom-	Nom	Nom+
- Language Type (complexity)	Nom	Nom	VHi
- Host Develop. System Complexity	Nom	Nom	Nom
- Application Class Complexity	Nom-	Nom	Nom+
- Practices & Procedures Complexity	Nom	Nom	Nom
+ TARGET ENVIRONMENT	Nom	Nom+	Hi-
- Special Display Requirements	Hi-	Hi	VHi
- Memory Constraints	Hi-	Hi	Hi+
- Time Constraints	Nom	Nom+	Hi-
- Real Time Code	Nom	Nom+	Hi
- Target System Complexity	Nom	Nom	Nom
- Target System Volatility	Hi-	Hi	Hi+
- Security Requirements	Nom	Nom	Nom
+ SCHEDULE			

Inputs

	Least	Likely	Most
- Required Schedule (Calendar Mos)		0.00	
- Start Date for Requirements Phase		7/31/93	
- Base Monetary Year		1993	
+ STAFFING			
- Maximum Staffing Rate Per Year		0.0	
- Maximum Total Staff Available		0.0	
- Maximum Effort Available (PMos)		0.00	
- Forced Overstaffing		Nom	
- Probability		50.00%	
+ S/W REQUIREMENTS ANALYSIS			
- Requirements Complete @ Contract		Low	
- Requirements Definition Formality	Nom	Nom	Hi
- Requirements Effort After Baseline		YES	
+ S/W TO S/W INTEGRATION			
- CSCIs Concurrently Integrating		2	
- Integration Organizations Involved		0	
- External Interfaces Among CSCIs		0	
+ S/W TO H/W INTEGRATION			
- Hardware Integration Level	Nom	Nom	Nom
- Unique Hardware Interfaces		0	
+ OTHER ADD-ONS			
+ SOFTWARE MAINTENANCE			
- Years of Maintenance		0	
- Separate Sites		1	
- Maintenance Growth Over Life		23.00%	
- Personnel Differences	Low	Nom-	Nom
- Development Environment Differences	Nom	Nom	Nom+
- Annual Change Rate		11.00%	
- Maintain Total System		YES	
+ ESTIMATE TO COMPLETE			
+ AVERAGE PERSONNEL COSTS		14,200	

Activity

Activity		Schedule Months	Person Months	Person Hours	Cost
System Concept		0.00	0.00	0	0
Cumulative		0.00	0.00	0	0
System Requirements Design		3.72	10.21	1,552	145,032
Cumulative	11/21/93	3.72	10.21	1,552	145,032
S/W Requirements Analysis		4.38	29.97	4,556	425,625
Cumulative	4/02/94	8.10	40.19	6,108	570,657
Preliminary Design		5.61	61.84	9,400	878,185
Cumulative	9/21/94	13.71	102.03	15,509	1,448,842
Detailed Design		6.45	103.73	15,766	1,472,922
Cumulative	4/04/95	20.16	205.76	31,275	2,921,764
Code & CSU Test		7.57	162.41	24,686	2,306,210
Cumulative	11/21/95	27.73	368.17	55,961	5,227,974
CSC Integrate & Test		7.57	193.91	29,474	2,753,521
Cumulative	7/10/96	35.30	562.08	85,436	7,981,495
CSCI Test		0.84	22.30	3,389	316,622
Cumulative	8/04/96	36.14	584.37	88,825	8,298,117
System Integrate Thru OT&E		5.15	135.03	20,525	1,917,470
Cumulative	1/08/97	41.29	719.41	109,350	10,215,588
Maintenance / Op Support		0.00	0.00	0	0
Cumulative	1/08/97	41.29	719.41	109,350	10,215,588

SEER-SEM (TM) Software Schedule, Cost & Risk Estimation Version 3.21
Project : Test Case 7/31/93
CSCI : CSCI 2 4:20:51 PM

Quick Estimate

Schedule Months	46.28
Effort Months	1,012.99
Base Year Cost	14,384.51K
Constraint	MIN TIME
Phases Included	REQ + FSI + STE

Inputs

	Least	Likely	Most
+ EFFECTIVE LINES	80,000	80,000	80,000
- New Lines of Code	80,000	80,000	80,000
+ PRE-EXISTS, NOT DESIGNED FOR REUSE	0	0	0
- Pre-existing lines of code	0	0	0
- Lines to be deleted in pre-exstg	0	0	0
- Redesign required	5.00%	10.00%	40.00%
- Reimplementation required	1.00%	5.00%	20.00%
- Retest required	10.00%	40.00%	100.00%
+ PRE-EXISTS, DESIGNED FOR REUSE	0	0	0
- COMPLEXITY	VHi-	VHi	VHi+
+ PERSONNEL CAPABILITIES & EXPERIENCE	Nom-	Hi-	Hi+
- Analyst Capabilities	Low	Nom	Hi
- Analyst's Application Experience	Nom-	Nom	Nom+
- Programmer Capabilities	Nom	Hi	VHi
- Programmer's Language Experience	Hi	VHi	EHi
- Host Development System Experience	Nom	Hi	VHi
- Target System Experience	Nom	Hi	Hi
- Practices & Methods Experience	Nom	Hi	VHi
+ DEVELOPMENT SUPPORT ENVIRONMENT	Low+	Nom	Nom
- Modern Development Practices Use	Nom-	Nom	Nom+
- Automated Tools Use	Low	Nom	Nom
- Logon thru Hardcopy Turnaround	VLo	Low+	Hi
- Terminal Response Time	Low-	Hi-	Hi
- Multiple Site Development	Nom	Nom	Nom
- Resource Dedication	Nom	Nom	Nom
- Resource and Support Location	Nom	Nom	Nom
- Host System Volatility	Nom	Nom	Nom
- Practices & Methods Volatility	Nom	Nom	Nom
+ PRODUCT DEVELOPMENT REQUIREMENTS	Nom	Nom	Hi
- Requirements Volatility (Change)	Nom	Nom	Hi
- Specification Level - Reliability	Nom	Nom	Hi
- Test Level	Nom	Nom	Hi
- Quality Assurance Level	Nom	Nom	Hi
- Rehost from Development to Target	Nom	Hi	VHi+
+ PRODUCT REUSABILITY REQUIREMENTS			
- Reusability Level Required	Nom	Nom	Nom
- Software Impacted by Reuse	0.00%	0.00%	0.00%
+ DEVELOPMENT ENVIRONMENT COMPLEXITY	Nom	Nom	Nom+
- Language Type (complexity)	Hi	Hi	VHi
- Host Develop. System Complexity	Nom	Nom	Nom
- Application Class Complexity	Nom-	Nom	Nom+
- Practices & Procedures Complexity	Nom	Nom	Nom
+ TARGET ENVIRONMENT	Nom	Nom+	Hi-
- Special Display Requirements	Hi-	Hi	VHi
- Memory Constraints	Hi-	Hi	Hi+
- Time Constraints	Nom	Nom+	Hi-
- Real Time Code	Nom	Nom+	Hi
- Target System Complexity	Nom	Nom	Nom
- Target System Volatility	Hi-	Hi	Hi+
- Security Requirements	Nom	Nom	Nom
+ SCHEDULE			

Inputs

	Least	Likely	Most
- Required Schedule (Calendar Mos)		0.00	
- Start Date for Requirements Phase		7/31/93	
- Base Monetary Year		1993	
+ STAFFING			
- Maximum Staffing Rate Per Year		0.0	
- Maximum Total Staff Available		0.0	
- Maximum Effort Available (PMos)		0.00	
- Forced Overstaffing		Nom	
- Probability		50.00%	
+ S/W REQUIREMENTS ANALYSIS			
- Requirements Complete @ Contract		Low	
- Requirements Definition Formality	Nom	Nom	Hi
- Requirements Effort After Baseline		YES	
+ S/W TO S/W INTEGRATION			
- CSCIs Concurrently Integrating		2	
- Integration Organizations Involved		0	
- External Interfaces Among CSCIs		0	
+ S/W TO H/W INTEGRATION			
- Hardware Integration Level	Nom	Nom	Nom
- Unique Hardware Interfaces		0	
+ OTHER ADD-ONS			
+ SOFTWARE MAINTENANCE			
- Years of Maintenance		0	
- Separate Sites		1	
- Maintenance Growth Over Life		23.00%	
- Personnel Differences	Low	Nom-	Nom
- Development Environment Differences	Nom	Nom	Nom+
- Annual Change Rate		11.00%	
- Maintain Total System		YES	
+ ESTIMATE TO COMPLETE			
+ AVERAGE PERSONNEL COSTS		14,200	

Activity

Activity	Schedule Months	Person Months	Person Hours	Cost
System Concept	0.00	0.00	0	0
Cumulative	0.00	0.00	0	0
System Requirements Design	4.17	14.38	2,186	204,219
Cumulative 12/05/93	4.17	14.38	2,186	204,219
S/W Requirements Analysis	4.91	42.21	6,415	599,320
Cumulative 5/03/94	9.07	56.59	8,601	803,539
Preliminary Design	6.29	87.08	13,237	1,236,568
Cumulative 11/10/94	15.36	143.67	21,838	2,040,107
Detailed Design	7.23	146.06	22,201	2,074,013
Cumulative 6/17/95	22.59	289.73	44,038	4,114,120
Code & CSU Test	8.49	228.69	34,760	3,247,362
Cumulative 3/04/96	31.08	518.41	78,799	7,361,482
CSC Integrate & Test	8.49	273.04	41,503	3,877,218
Cumulative 11/17/96	39.57	791.46	120,302	11,238,699
CSCI Test	0.94	31.40	4,772	445,834
Cumulative 12/16/96	40.51	822.85	125,074	11,684,534
System Integrate Thru OT&E	5.77	190.14	28,901	2,699,979
Cumulative 6/08/97	46.28	1,012.99	153,975	14,384,513
Maintenance / Op Support	0.00	0.00	0	0
Cumulative 6/08/97	46.28	1,012.99	153,975	14,384,513

Quick Estimate

Schedule Months	39.59
Effort Months	633.97
Base Year Cost	9,002.32K
Constraint	MIN TIME
Phases Included	REQ + FSI + STE

SEER-SEM (TM) Software Schedule, Cost & Risk Estimation Version 3.21
Project : Test Case 7/31/93
CSCI : CSCI 3 4:21:04 PM

Inputs

	Least	Likely	Most
+ EFFECTIVE LINES	45,000	45,000	45,000
- New Lines of Code	45,000	45,000	45,000
+ PRE-EXISTS, NOT DESIGNED FOR REUSE	0	0	0
- Pre-existing lines of code	0	0	0
- Lines to be deleted in pre-exstg	0	0	0
- Redesign required	5.00%	10.00%	40.00%
- Reimplementation required	1.00%	5.00%	20.00%
- Retest required	10.00%	40.00%	100.00%
+ PRE-EXISTS, DESIGNED FOR REUSE	0	0	0
- COMPLEXITY	VHi-	VHi	VHi+
+ PERSONNEL CAPABILITIES & EXPERIENCE	Low-	Nom	Hi-
- Analyst Capabilities	Low	Nom	Hi
- Analyst's Application Experience	Nom-	Nom	Nom+
- Programmer Capabilities	Low	Nom	Hi
- Programmer's Language Experience	Low-	Nom	Hi
- Host Development System Experience	Low-	Nom	Hi
- Target System Experience	Low-	Nom	Hi
- Practices & Methods Experience	Low-	Nom	Hi
+ DEVELOPMENT SUPPORT ENVIRONMENT	Low+	Nom	Nom
- Modern Development Practices Use	Nom-	Nom	Nom+
- Automated Tools Use	Low	Nom	Nom
- Logon thru Hardcopy Turnaround	VLo	Low+	Hi
- Terminal Response Time	Low-	Hi-	Hi
- Multiple Site Development	Nom	Nom	Nom
- Resource Dedication	Nom	Nom	Nom
- Resource and Support Location	Nom	Nom	Nom
- Host System Volatility	Nom	Nom	Nom
- Practices & Methods Volatility	Nom	Nom	Nom
+ PRODUCT DEVELOPMENT REQUIREMENTS	Nom	Nom	Hi
- Requirements Volatility (Change)	Nom	Nom	Hi
- Specification Level - Reliability	Nom	Nom	Hi
- Test Level	Nom	Nom	Hi
- Quality Assurance Level	Nom	Nom	Hi
- Rehost from Development to Target	Nom	Hi	VHi+
+ PRODUCT REUSABILITY REQUIREMENTS			
- Reusability Level Required	Nom	Nom	Nom
- Software Impacted by Reuse	0.00%	0.00%	0.00%
+ DEVELOPMENT ENVIRONMENT COMPLEXITY	Nom-	Nom	Nom+
- Language Type (complexity)	Nom	Nom	VHi
- Host Develop. System Complexity	Nom	Nom	Nom
- Application Class Complexity	Nom-	Nom	Nom+
- Practices & Procedures Complexity	Nom	Nom	Nom
+ TARGET ENVIRONMENT	Nom	Nom+	Hi-
- Special Display Requirements	Hi-	Hi	VHi
- Memory Constraints	Hi-	Hi	Hi+
- Time Constraints	Nom	Nom+	Hi-
- Real Time Code	Nom	Nom+	Hi
- Target System Complexity	Nom	Nom	Nom
- Target System Volatility	Hi-	Hi	Hi+
- Security Requirements	Nom	Nom	Nom
+ SCHEDULE			

Inputs

	Least	Likely	Most
- Required Schedule (Calendar Mos)		0.00	
- Start Date for Requirements Phase		7/31/93	
- Base Monetary Year		1993	
+ STAFFING			
- Maximum Staffing Rate Per Year		0.0	
- Maximum Total Staff Available		0.0	
- Maximum Effort Available (PMos)		0.00	
- Forced Overstaffing		Nom	
- Probability		50.00%	
+ S/W REQUIREMENTS ANALYSIS			
- Requirements Complete @ Contract		Low	
- Requirements Definition Formality	Nom	Nom	Hi
- Requirements Effort After Baseline		YES	
+ S/W TO S/W INTEGRATION			
- CSCIs Concurrently Integrating		2	
- Integration Organizations Involved		0	
- External Interfaces Among CSCIs		0	
+ S/W TO H/W INTEGRATION			
- Hardware Integration Level	Nom	Nom	Nom
- Unique Hardware Interfaces		0	
+ OTHER ADD-ONS			
+ SOFTWARE MAINTENANCE			
- Years of Maintenance		0	
- Separate Sites		1	
- Maintenance Growth Over Life		23.00%	
- Personnel Differences	Low	Nom-	Nom
- Development Environment Differences	Nom	Nom	Nom+
- Annual Change Rate		11.00%	
- Maintain Total System		YES	
+ ESTIMATE TO COMPLETE			
+ AVERAGE PERSONNEL COSTS		14,200	

Activity

Activity	Schedule Months	Person Months	Person Hours	Cost
System Concept	0.00	0.00	0	0
Cumulative	0.00	0.00	0	0
Sys n Requirements Design	3.56	9.00	1,368	127,807
Cumulative 11/16/93	3.56	9.00	1,368	127,807
S/W Requirements Analysis	4.20	26.41	4,015	375,075
Cumulative 1/25/94	7.76	35.41	5,383	502,882
Preliminary Design	5.38	54.50	8,284	773,886
Cumulative 9/04/94	13.14	89.91	13,667	1,276,768
Detailed Design	6.18	91.41	13,894	1,297,988
Cumulative 3/12/95	19.32	181.32	27,561	2,574,757
Code & CSU Test	7.26	143.12	21,754	2,032,310
Cumulative 10/16/95	26.58	324.44	49,315	4,607,066
CSC Integrate & Test	7.26	170.38	25,974	2,426,495
Cumulative 5/26/96	33.84	495.32	75,289	7,033,561
CSCI Test	0.81	19.65	2,987	279,018
Cumulative 6/19/96	34.65	514.97	78,275	7,312,579
System Integrate Thru OT&E	4.94	119.00	18,087	1,689,739
Cumulative 11/17/96	39.59	633.97	96,363	9,002,318
Maintenance / Op Support	0.00	0.00	0	0
Cumulative 11/17/96	39.59	633.97	96,363	9,002,318

Bibliography

1. Boehm, Barry W. Software Engineering Economics. Englewood Cliffs NJ: Prentice-Hall Inc., 1981.

2. - - - - - , and Philip N. Papaccio. "Understanding and Controlling Software Costs," IEEE Transactions on Software Engineering, 14: 1462 - 1477 (October 1988).

3. Ourada, Capt Gerald L. Software Cost Estimating Models: A Calibration, Validation, and Comparison. MS Thesis, AFIT/GSS/LSY/91D-11. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, December 1991 (AD-A246677).

4. Gurner, Capt Robert B. A Comparative Study of the Reliability of Function Point Analysis in Software Development Effort Estimation Models. MS Thesis, AFIT/GCA/LSY/91S-2. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, September 1991 (AD-A244179).

5. Daly, Capt Bryan A. A Comparison of Software Schedule Estimators. MS Thesis, AFIT/GCA/LSQ/90S-1. School of Systems and Logistics, Air Force Institute of Technology, Wright-Patterson AFB OH, September 1990 (AD-A229532).

6. IIT Research Institute. Test Case Study: Estimating the Cost of Ada Software Development. Lanham MD: April 1989.

7. Office of the Assistant Secretary, Washington DC. Memorandum: Air Force Software Estimating Models. ALMAJCOM-FOA/CV, Direct Reporting Unit Commanders, and Air Force Program Executive Officers, 11 May 1992.

8. Stewart, Rodney D. and Richard M. Wyskida. Cost Estimator's Reference Manual. New York NY: John Wiley & Sons, Inc., 1987.

9. Analytic Sciences Corporation, The. The AFSC Cost Estimating Handbook. Reading MA: prepared for USAF, Air Force Systems Command (AFSC), 1986.

10. Department of Defense. Military Standard. Defense System Software Development. DoD-STD-2167A. Washington DC: GPO, 29 February 1988.

11. Schwenke, Robert S. Class handout, COST 291, Introduction to Cost Analysis. School of Logistics and Acquisition Management, Air Force Institute of Technology, Wright-Patterson AFB OH, Summer Short Quarter 1992.

12. SASET. Version 3.0, IBM, disk. Computer software tutorial. Martin Marietta Corporation, Denver CO, 1990.

13. Ferens, Daniel V. "New Perspectives in Software Logistics Support," Logistics Spectrum, 4-8 (Spring 1992).

14. Putnam, Lawrence H. Measures for Excellence: Reliable Software on Time, Within Budget. Englewood Cliffs NJ: Prentice-Hall Inc., 1992.

15. Schlender, Brenton R. "How to Break the Software Logjam," Fortune, 120: 100-108 (September 1989).

16. Brooks, Frederick P. Jr. The Mythical Man-Month. Menlo Park CA: Addison-Wesley, 1975.

17. - - - - - . "No Silver Bullet: Essence and Accidents of Software Engineering," Computer, 10-19 (April 1987).

18. Huff, Karen E. et al. "Quantitative Models for Managing Software Development Processes," Software Engineering Journal, 17-23 (January 1986).

19. SEER User's Manual, Galorath Associates Incorporated, Los Angeles CA. March 1991.

20. Kile, Raymond L. REVIC Software Cost Estimating Model User's Manual, version 9.0, April 1991.

21. Symons, Charles R. Software Sizing and Estimating: MK II FPA (Function Point Analysis). Chichester, England: John Wiley & Sons, Ltd., 1991.

22. Ferens, Daniel V. and Robert B. Gurner. "An Evaluation of Three Function Point Models for Estimation of Software Effort," NAECON Conference, May 1992.

23. Bruce, Phillip and Sam M. Pederson. The Software Development Project: Planning and Management. New York NY: John Wiley & Sons, Inc., 1982.

24. Davis, Alan M. et al. "A Strategy for Comparing Alternative Software Development Life Cycle Models," IEEE Transactions on Software Engineering, 14: 1453-1461 (October 1988).

25. Rook, Paul. "Controlling Software Projects," Software Engineering Journal, 7-16 (January 1986).

26. Silver, Dr. Aaron et al. SASET User's Guide, Version Number 2.0, Publication R-0330-90-2, Naval Center for Cost Analysis, Department of the Navy, Washington DC: February 1990.

27. General Electric Company. PRICE-S Reference Manual. Moorestown NJ: GE-Price Systems, December 1989.

28. - - - - - . The Central Equations of the PRICE Software Cost Model. Moorestown NJ: GE-Price Systems, undated.

29. Maness, Richard. Senior Software Engineer, Martin Marietta Corporation, Denver CO. Telephone interviews. 4 May through 29 July 1993.

30. Otte, James. Software Engineer, PRICE Systems, Dayton OH. Personal interview. 23 June 1993.

31. Otte, James. Class handout, IMGT 677, Quantitative Management of Software. School of Logistics and Acquisition Management, Air Force Institute of Technology, Wright-Patterson AFB OH, Fall Quarter 1992.

32. McRitchie, Karen. Software Technician, SEER Technologies Division, Galorath Associates, Inc., Los Angeles CA. Telephone interview. 3 July 1993.

33. Galloraih, Daniel. SEER-SEM Model developer. SEER Technologies Division, Galorath Associates, Inc., Los Angeles CA. Facsimile transmission, 28 July 1993.

34. Kile, Raymond. REVIC Model Developer. Telephone interview. 28 June through 29 July 1993.

35. Otte, James. Software Engineer, PRICE Systems, Dayton OH. Untitled internal report regarding relationship between PRICE-S and checklists completed by the Software Engineering Institute. Undated.

36. Lewis, Richard. Software Technician, SEER Technologies Division, Galorath Associates, Inc., Los Angeles CA. Telephone interview. 9 July 1993.

37. SEER User's Manual, SEER Technologies Division, Galorath Associates Incorporated, Los Angeles CA. November 1992.

38. Greve, Alan R., and others. The REVIC Advisor (REVAD): An Expert System Preprocessor to a Parametric Software Cost Estimating Model. Defense Logistics Agency, Cameron Station VA, September 1991.

39. Fugate, Carol. Software Technician, PRICE Systems, Dayton OH. Telephone interview. 30 July 1993.

40. Rowland, Sydney. A Comparative Analysis of the GE PRICE-S and the CEI System-4 Cost Estimating Models. Wright-Patterson AFB OH: ASD/YTFF, May 1992.

VITA

Captain George A. Coggins (Andy) was born on 28 May 1965 in Okinawa, Japan. He graduated from high school in Jacksonville, Arkansas in 1983 and received the degree of Bachelor of Science from the United States Air Force Academy in 1987. After graduation, he served five years as an Audit Team Leader at the Area Audit Office at Kirtland AFB, New Mexico. Captain Coggins then entered the School of Logistics and Acquisition Management, Air Force Institute of Technology, in May 1992.

Permanent Address: 607 Caribbean Way
Niceville, FL 32578

VITA

Captain Roy C. Russell (Chris) was born on 4 July 1966 in Ada, Oklahoma. He graduated from Peoria High School, Arizona in 1984 and received the degree of Bachelor of Science from the United States Air Force Academy in 1988. After graduation, he served four years as an Audit Team Leader at the Area Audit Office at Cannon AFB, New Mexico. He then entered the School of Logistics and Acquisition Management, Air Force Institute of Technology, in May 1992.

Permanent Address: 7620 West Beryl Ave.
Peoria, AZ 85345

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1993	Master's Thesis

4. TITLE AND SUBTITLE
SOFTWARE COST ESTIMATING MODELS: A COMPARATIVE STUDY OF WHAT THE MODELS ESTIMATE

5. FUNDING NUMBERS

6. AUTHOR(S)
George A. Coggins, Captain, USAF
Roy C. Russell, Captain, USAF

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
WPAFB OH 45433-6583

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT/GCA/LAS/93S-4

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Cost Analysis Agency
John B. Donald
1111 Jefferson Davis Hwy, Suite 403
Arlington VA 22202

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This effort developed a consolidated document which highlights and examines differences in definitions, assumptions, and methodologies used by the REVIC, SASET, PRICE-S, and SEER-SEM cost models. The following research questions were investigated: (1) What differences exist between the cost models? (2) How do these differences impact the resulting estimates? (3) To what degree can we explain and adjust for known differences between the cost models?

Seven specific areas were addressed: (1) software development phases, (2) development activities and cost elements, (3) source lines of code and language differences, (4) key model attributes and key cost drivers, (5) implications of project size on model output, (6) impact of schedule compression and extensions, and (7) distinctive characteristics of the model data bases.

A hypothetical baseline test case was developed to determine if users could explain and adjust for known differences. It is the researchers' opinion that the underlying equations and model assumptions are so dissimilar that objective normalization efforts are virtually impossible for the average model user.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Cost Estimates, Cost Models, Software (Computers), Comparison, Models	173
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	Unlimited

AFIT Control Number AFIT/GCA/LAS/93S-4

AFIT RESEARCH ASSESSMENT

The purpose of this questionnaire is to determine the potential for current and future applications of AFIT thesis research. Please return completed questionnaires to: DEPARTMENT OF THE AIR FORCE, AIR FORCE INSTITUTE OF TECHNOLOGY/LAC, 2950 P STREET, WRIGHT PATTERSON AFB OH 45433-7765

1. Did this research contribute to a current research project?

   a. Yes b. No

2. Do you believe this research topic is significant enough that it would have been researched (or contracted) by your organization or another agency if AFIT had not researched it?

   a. Yes b. No

3. The benefits of AFIT research can often be expressed by the equivalent value that your agency received by virtue of AFIT performing the research. Please estimate what this research would have cost in terms of manpower and/or dollars if it had been accomplished under contract or if it had been done in-house.

   Man Years _____ $ _____

4. Often it is not possible to attach equivalent dollar values to research, although the results of the research may, in fact, be important. Whether or not you were able to establish an equivalent value for this research (3, above) what is your estimate of its significance?

   a. Highly Significant b. Significant c. Slightly Significant d. Of No Significance

5. Comments

_____ _____

Name and Grade Organization

_____ _____

Position or Title Address